Computer Organization and Architecture

UNIT 2

Chapter 1

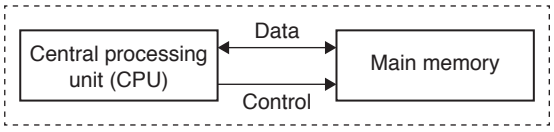
Machine Instructions, Addressing Modes

LEARNING OBJECTIVES

- *Computer*
- *Computer system*
- *Computer component*
- *Machine instruction*
- *Instruction types*
- *Types of operands*
- *Types of operations*
- *Procedure call instruction*
- *Addressing modes*
- *Computer performance*

Computer

A computer is a data-processing machine which is operated automatically under the control of a list of instructions (called a program) stored in its main memory.



Computer System

- A computer system consists usually of a computer and its peripherals.
- Computer peripherals include input devices, output devices and secondary memories.

Computer Architecture: Computer Architecture refers to those attributes of a system visible to a programmer, i.e., the attributes that have direct impact on the logical execution of a program.

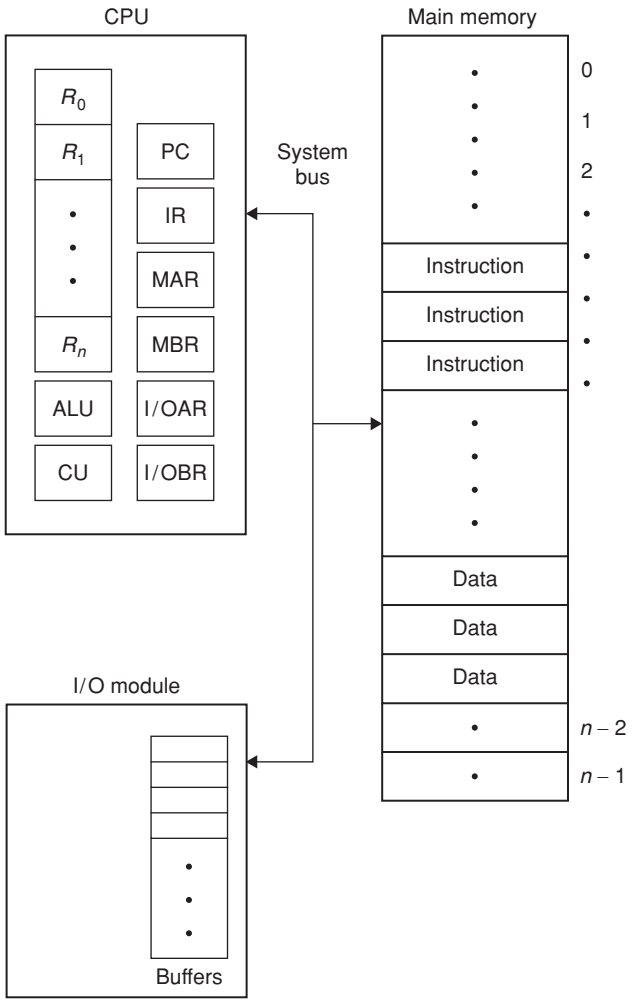
Example: Whether a computer will have a multiply instruction or not.

Computer Organization: Computer organization refers to operational units and their interconnections that realize the architectural specifications.

Example: Whether the multiply instruction will be implemented by a special multiply unit or by a mechanism that makes repeated use of the add unit of the system.

Computer Components

- $R_1, R_2 \ldots R_n$: General Purpose Registers.
- **PC:** Program counter. Holds address of next instruction to be executed. PC = PC + I (I = instruction length)
- **IR:** Instruction Register. It holds the instruction which is fetched from memory.
- **MAR:** Memory Address Register: MAR specifies the address in memory for the next read or write.
- **MBR:** Memory Buffer Register. It contains the data to be written into memory or receives the data read from memory.
- **Input–outputAR:** Input–output Address Register. It specifies a particular input–output device.
- **Input–outputBR:** Input–output Buffer Register. Used for the exchange of data between an input–output module and the CPU.
- **ALU:** Arithmetic and Logic Unit. Used to perform arithmetic and logical operations.



- **CU:** Control Unit. It causes operations to happen within the processor. Also generates timing signals.
- **Memory:** It consists of set of locations, defined by sequential numbered address.
- **Input–output module:** Transfer data from external device to CPU and memory and vice versa.
- **System bus:** A bus that connects major computer components is called a system bus.

Machine Instructions

- The operation of the CPU is determined by the instructions it executes. These instructions are called machine instructions or computer instructions.
- The collection of different instructions that the CPU can execute is referred to as the CPU's instruction set.

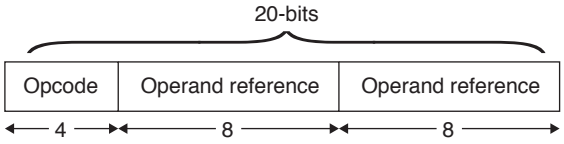
Elements of Machine Instructions

Each instruction must contain the information required by the processor for execution. The elements of a machine instruction are

1. Operation code: Specifies the operation
2. Source operand reference: Inputs for operation.
3. Result operand reference
4. Next instruction reference

Instruction representation

- Each instruction is represented by a sequence of bits.
- Example: 20-bit instruction format:



Instruction Types

Number of addresses

Most of the instructions have one, two or three operand addresses, with the address of next instruction being implicit.

(i) 3-address instructions: Computers with 3-address instruction formats can use each address field to specify either a processor register or a memory operand.

Example: 3-address instruction format for the evaluation of $X = (P + Q) \times (R + S)$ is

ADD R_1, P, Q
ADD R_2, R, S
MUL X, R_1, R_2.

Here R_1, R_2 are processor registers.

Advantage: Shorter programs when evaluating arithmetic expressions.

Disadvantage: The binary coded instructions required too many bits to specify three addresses.

(ii) 2-address instructions: These are most common in commercial computers. Each address field can specify either a processor register or a memory word.

Example: For evaluating $X = (P + Q) \times (R + S)$,

The 2-address instructions are

MOV R_1, P
ADD R_1, Q
MOV R_2, R
ADD R_2, S
MUL R_1, R_2
MOV X, R_1.

(The first symbol of instruction is both source and destination)

(iii) One-address instructions: Use an implied accumulator (AC) register for all data manipulations.

Example: 1-address instructions to evaluate $R = (P + Q) \times (R + S)$.

LOAD P
ADD Q

STORE T
LOAD R
ADD S
MUL T
STORE X.

Here 'T' is a temporary memory location required to store the intermediate result.

(iv) Zero-address instructions: A stack organized computer does not use an address field for the instructions ADD and MUL. The push and pop instructions require an address field to specify the operand that communicates with the stack.

Example: Zero-address instructions for the evaluation of $X = (P + Q) \times (R + S)$

PUSH P
PUSH Q
ADD
PUSH R
PUSH S
ADD
MUL
POP X.

(v) RISC instructions: The instruction set of a reduced instruction set computer (RISC) processor is restricted to the use of load and store instruction when communicating between memory and CPU. All other instructions are executed with in the register of the CPU without referring to memory.

Example: RISC instruction to evaluate,
$X = (P + Q) \times (R + S)$

LOAD R_1, P
LOAD R_2, Q
LOAD R_3, R
LOAD R_4, S
ADD R_1, R_1, R_2
ADD R_3, R_3, R_4
MUL R_1, R_1, R_3
STORE X, R_1

Types of Operands

Machine instructions operate on data. The most important general categories of data are

- Addresses
- Numbers
- Characters
- Logical data.

Types of Operations

The number of different opcodes varies widely from machine to machine. A useful and typical categorization is the following

1. Data transfer
2. Arithmetic
3. Logic
4. Conversion
5. Input–output
6. System control
7. Transfer of control

(i) Data transfer operations: This type of instructions transfers data from one location to another.

Example: move, store, load, exchange, clear, set, push, pop.

(ii) Arithmetic operations: Perform some function in ALU.

Example: add, subtract, multiply, divide, absolute, negate, increment, decrement

(iii) Logical operations: Perform some logical operation in ALU and set condition codes and flags.

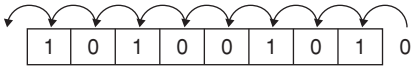
Example: AND, OR, NOT, EX-OR, Test, Compare, set control variables, shift, Rotate.
Let R_1 = 10100101, R_2 = 00001111 then
(R_1) AND (R_2) = 00000101.
AND is also called mask operation.
(R_1) OR (R_2) = 10101111.
NOT (R_1) = 01011010.
(R_1) EX-OR (R_2) = 10101010.

(iv) Shifting and rotating operations: The operations are:

(a) Logical left shift:

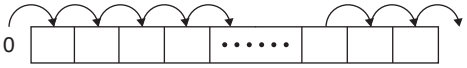
Here the bits of a word are shifted left. The left most bits is lost and 0 is shifted in right most bit position (i.e., bit empty).

Example: R_1 = 1010 0101
Logical left shift R_1:

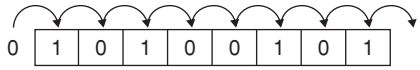


After left shift R_1 = 0100 1010.

(b) Logical right shift: Here the bits of a word are shifted right. The right most bit lost and '0' is shifted in left most bit position.

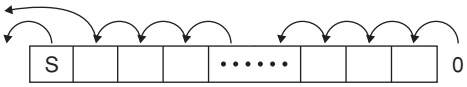


Example: R_1 = 1010 0101
Logical right shift R_1 = 0101 0010

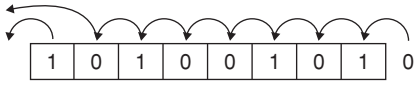


Logical shift operations are useful primarily for isolating fields within a word and also used to displace unwanted information.

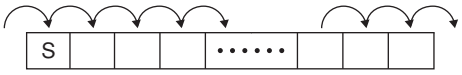
(c) Arithmetic left shift: Arithmetic shift operation treats the data as a signed integer and does not shift the sign bit. In Arithmetic left shift, a logical left shift is performed on all bits but the sign bit, which is retained.



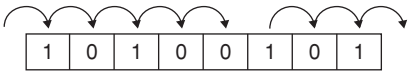
Example: $R_1 = 1010\ 0101$
Arithmetic Left shift $R_1 = 1100\ 1010$.



(d) Arithmetic right shift: Here, the sign bit is replicated into the bit position to its right.



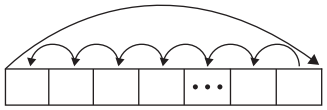
Example: $R_1 = 1010\ 0101$
Arithmetic Right shift $R_1 = 1101\ 0010$



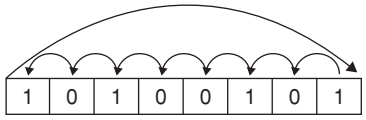
Notes:

1. With numbers in 2's complement form, a right arithmetic shift corresponds to a division by 2, with truncation for odd numbers.
2. Both arithmetic left shift and logical left shift correspond to a multiplication by 2 when there is no overflow.

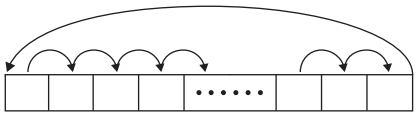
(e) Left rotate (Cyclic left shift): Rotate operations preserve all the bits being operated on. Here the bits from LSB will move one bit position to the left and MSB will placed in LSB position.



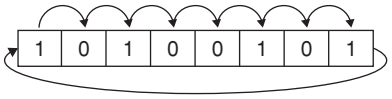
Example: $R_1 = 1010\ 0101$
Left Rotate $R_1 = 0100\ 1011$



(f) Right rotate (Cyclic right shift): Here the bits from MSB will be shifted to one bit position right and LSB is placed in MSB.



Example: R_1: 1010 0101
Right Rotate $R_1 = 1101\ 0010$.



(v) Transfer of control: This type of operations updates the program counter. Used for subroutine call/return, manage parameter passing and linkage.

Example: Jump, jump unconditional, return, execute, skip, skip conditional, Halt, Wait, NOP, etc.

(vi) Input–output operations: These are used to issue a command to input–output module.

Example: input, output, start input–output, test input–output etc.

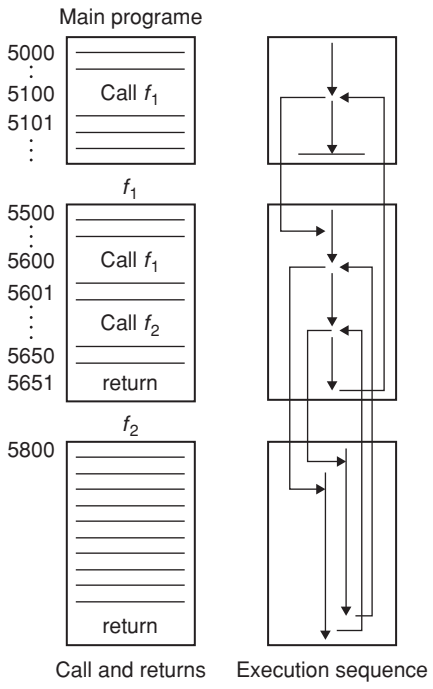
(vii) Conversion operations: These are similar to arithmetic and logical operations. May also involve special logic to perform conversion.

Procedure Call Instruction

A procedure is a self-contained computer program that is incorporated into a large program. It allows us to use the same piece of code many times. The procedure mechanism involves two basic instructions:

1. A call instruction that branches from the present location to the procedure.
2. A return instruction that returns from the procedure to the place from where it was called.

Example:



We can call a procedure from a variety of points, so the processor must somehow save the return address to return

appropriately. We can store the return address in the following places.

1. Register
2. Start of called procedure
3. Top of stack.

If the register approach is used, call X causes the following actions:

$$RN \leftarrow PC + L$$
$$PC \leftarrow X$$

Where RN is a used to store return address, PC is the program counter and L is instruction length.

To store the return address at the start of the procedure for call X, the following tasks required.

$$X \leftarrow PC + L$$
$$PC \leftarrow X + 1$$

We can pass the parameters using registers or store in memory after call instruction or use stack.

Example 1: Consider the following program fragment in the assembly language of a certain hypothetical processor. The processor has three 8-bit general purpose registers R_1, R_2, R_3.

Instruction	Meaning
X: CMP R_1,0	Compare R_1 and 0, set flags appropriately in status register.
JZ *Z*	Jump if zero to target *Z*.
MOV R_2, R_1	Copy contents of R_1 to R_2.
SHR R_1	Shift Right R_1
SHL R_1	Shift left R_1
CMP R_2, R_1	Compare R_2 and R_1 and set flag in status register
JZ *Y*	Jump if zero to *Y*.
INC R_3	Increment R_3
Y: SHR R_1	Shift Right R_1 by 1-bit
JMP *X*	Jump to *X*
Z: ...	

Let R_1, R_2, R_3 contain the values 3, 0, 0 respectively. What are the final values of R_1, R_2, R_3 when control reaches *Z*?

(A) 0, 0, 0 (B) 0, 1, 2
(C) 0, 1, 1 (D) 0, 2, 1

Solution: (B)

$R_1 = 0000\ 0011$
$R_2 = 0000\ 0000$
$R_3 = 0000\ 0000$

CMP R_1, 0, As $R_1 \neq 0 \Rightarrow$ Zero flag = 0.
Jump to *Z* if $Zf = 1$; but $Zf = 0$
MOV R_2, R_1; $R_2 \leftarrow R_1$ i.e., $R_2 = 0000\ 0011$
SHIFT right R_1; $R_1 = 0000\ 0011$
$\Rightarrow$ shr(R_1) = 00000001
SHIFT left R_1; $R_1 = 0000\ 0001 \Rightarrow$ shl (R_1)
$= 0000\ 0010$
Compare R_2, R_1; $R_1 \neq R_2 \Rightarrow ZF = 0$
As Zero flag is not set, increment R_3: $R_3 = 0000\ 0001$
Shift Right R_1; i.e., 0000 0001.
Jump to *X*.
Compare R_1, 0; As $R_1 \neq 0 \Rightarrow ZF = 0$.
Move R_2, R_1; $R_2 \leftarrow 0000\ 0001$.
Shift right R_1; $R_1 = 0000\ 0000$.
Shift left R_1; $R_1 = 0000\ 0000$.
Compare R_2, R_1; $R_1 \neq R_2 \Rightarrow ZF = 0$.
Increment R_3; $R_3 = 0000\ 0010$.
Shift Right R_1; 0000 0000.
Jump to *X*.
Compare R_1, 0; As $R_1 = 0 \Rightarrow ZF = 1$
As $ZF = 1$, jump to *Z*.
$\therefore R_1 = 0; R_2 = 1; R_3 = 2$.

Addressing Modes

The different ways in which the location of an operand is specified in an instruction are referred to as addressing modes.

Computers use addressing mode techniques for the purpose of accommodating the following provisions:

1. Facilitates pointers to memory.
2. Facilitates counters for loop control
3. Facilitates indexing of data
4. Facilitates program relocation.
5. Reduce the number of bits in the addressing field of the instruction.

The most common addressing techniques are

i. Implied mode
ii. Immediate mode
iii. Direct mode
iv. Indirect mode
v. Register mode
vi. Register Indirect mode
vii. Auto-increment or Auto-decrement mode
viii. Displacement mode
- PC relative mode
- Indexed mode
- Base register mode

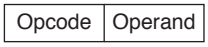
(i) Implied mode: Operands are specified implicitly in the definition of the instruction.

Example: CPL (complement accumulator)

Here operand in accumulator is implied in the definition of instruction.

- All register-reference instructions that use an accumulator are implied mode instructions.
- Zero-address instructions in a stack-oriented computer are implied-mode instructions.

(ii) Immediate mode: The operand is specified in the instruction itself. Instruction format in immediate mode is

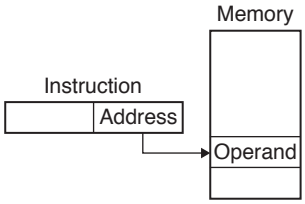


Example: Move A, 50.

- These are useful for initializing registers to constant value or to set initial values of variables.
- No memory reference is required other than the instruction fetch.
- The size of number is restricted to the size of the address (operand) field.

(iii) Direct mode:

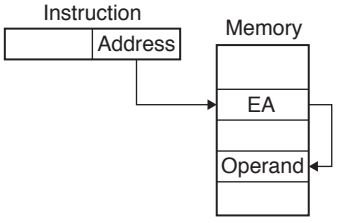
- Here the address of the operand is equal to the address part of the instruction.



- Required only one memory reference and no special calculation required.
- Limitation is limited address space.

(iv) Indirect mode:

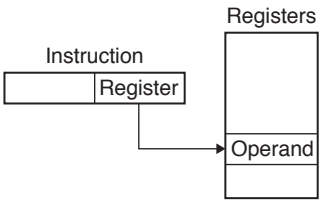
- The address field of the instruction gives the address of the operand which is stored in memory.
- The advantage of this approach is that for a word length of N, an address space of 2^N is available.
- The disadvantage is that the instruction execution requires two memory references to fetch the operand.



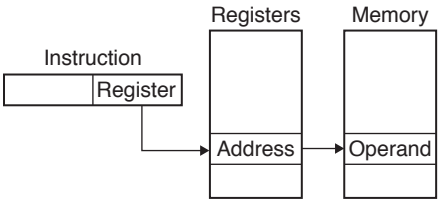
(Here EA is effective address of operand)

(v) Register mode:

- Here the operands are in registers that reside within the CPU.
- Only small address field required in instructions.
- No time consuming memory references are required.
- Address space is very limited.

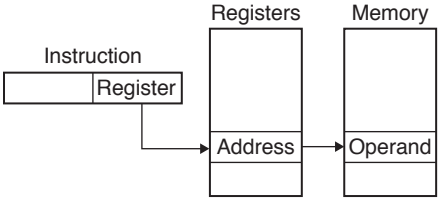


(vi) Register indirect mode: In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in memory. Address field of the instruction uses fewer bits to select a register than would have required to specify a memory address directly.



Effective address: The effective address is defined to be the memory address obtained from the computation based on the addressing mode, consists the actual address of the operand.

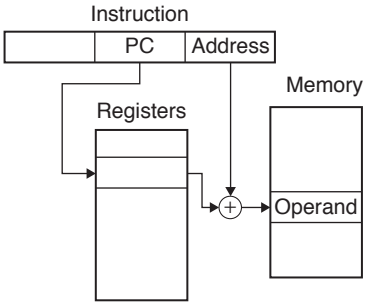
(vii) Auto increment and auto decrement mode: This is similar to register indirect mode except that the register is incremented or decremented after (or before) its value is used to access memory.



- After fetch operand, increment or decrement address.
- Used to access table of data.

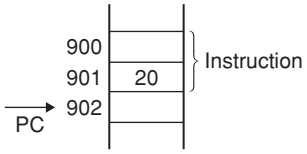
(viii) PC-relative mode: Here the content of the program counter is added to the address part of the instruction to get the effective address.

The address part of the instruction is usually a signed number which can be either positive or negative. The effective address will be a displacement relative to address of the inst ruction.



Effective address = PC + address part

Example: Let PC = 900 and address part of the 2-word instruction = 20.

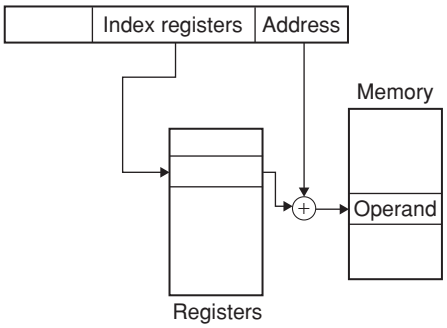


The instruction at location 900 is read memory during fetch phase and the PC will be incremented by instruction length i.e., 2. Then PC = 902.

∴ The effective address using PC-relative = 902 + 20 = 922.

- This addressing mode is used with branch-type instructions.
- Requires shorter address field.

(ix) Indexed mode: Here the content of the index register is added to the address part of the instruction to obtain the effective address. The address field of the instruction defines the beginning address of a data array in memory. The distance between the beginning address and the address of the operand is the index value stored in the index register, is a positive displacement from the address.

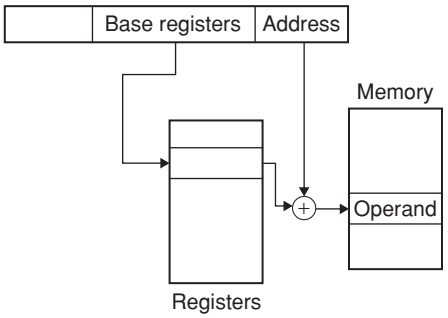


Effective Address = Index Register + Address part of instruction.

This approach is opposite to the interpretation of base-register addressing. This is used to provide an efficient mechanism for performing iterative operations. To store an array using indexed mode, the address part consists of the address of first element of array and the index register specifies the index value.

Example: Let address part of instruction = 300
Index register = 5 and each element of array requires 2 bytes then address of 5th element = 300 + 5 × 2 = 310.

(x) Base Register mode: Here the content of a base register is added to the address part of the instruction to obtain the effective address. Base register has a base address and the address field of the instruction gives a displacement relative to the base address.



This addressing mode is used in computers to facilitate the relocation of programs in memory, i.e., when programs and data are moved from one segment of memory to another, as required in multiprogramming system, the address values of instructions must reflect this change of position. With a base register, the displacement values of instructions do not have to change. Only the value of the base register requires updating to reflect the beginning of a new memory segment.

Example 2: If base register is 200 and address part of the instruction is 31, then effective address = 200 + 31 = 231.

If the program's base address is changed from 200 to 400, then new effective address will be 400 + 31 = 431.

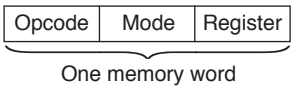
Example 3: Match the following:

	LIST I		LIST II
P.	P[i] = Q[i];	**1.**	Indexed mode
Q.	while(i++);	**2.**	Immediate mode
R.	int i = 10;	**3.**	Auto increment mode

(A) P – 1, Q – 2, R – 3 (B) P – 2, Q – 3, R – 1
(C) P – 1, Q – 2, R – 2 (D) P – 1, Q – 2, R – 2

Solution: (C)
Array indexing uses indexed mode. For increment operations use Auto increment mode. To initialize variables use immediate mode.

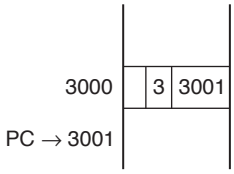
Example 4: The instruction format of a CPU is



Mode and Register together specifies the operand. Register specifies a CPU register and mode specifies an addressing mode. Let mode = 3, specifies that the register contains the address of the operand, after fetching the operand, the contents of register are incremented by 1. An instruction at memory location 3000 specifies mode = 3 and register refers to program counter (PC). Then what is the address of the operand?

(A) 3000 (B) 3001
(C) 3002 (D) Data insufficient

Solution: (B)



∴ Address of operand = 3001

Example 5: Consider the following machine instruction:

$$\text{MUL } P[R_0], @Q$$

The first operand (destination) '$P[R_0]$' uses indexed addressing mode with R_0 as the index register. The second operand (source) '$@Q$' uses indirect addressing mode. P and Q are memory addresses residing at the second and third words respectively. The first word of instruction specifies the opcode, the index register designation, source and destination addressing modes. During the execution of MUL, the result is stored in destination. How many memory cycles needed during the execution cycle of the instruction?

(A) 3 (C) 5
(B) 4 (D) 6

Solution: (C)

The first operand $P[R_0]$ uses indexed mode. So it requires two memory references: on reference to the address part and next one to obtain the operand. The second operand $@Q$ uses indirect addressing mode, so it requires two memory references, one to obtain the address, and the second, to obtain operand. Finally one more memory reference required to store result. Total five references.

Computer Performance

Response time: The time between the start and completion of a task. This is also referred as execution time.

Throughput: The total amount of work done in a given time.

Performance can be defined as Performance

$$\text{Pertermance} = \frac{1}{\text{Execution time}}$$

CPU execution time or CPU time: This is the time the CPU spends computing for the task and does not include time spent waiting for input–output or running other programs.

CPU time can be divided into

1. User CPU time
2. System CPU time

User CPU time: CPU time spent in the program.

System CPU time: CPU time spent in the operating system performing tasks on behalf of the program.

Clock cycle: Computers are constructed using a clock that determines when events take place in the hardware. These discrete time intervals are called clock cycles.

Clock period: The length of each clock cycle.

Clock rate: Inverse of the clock period.

CPU execution time for a program = CPU clock cycles for a program * clock cycle time

$$= \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$

CPU clock cycles = Instructions for a program * Average clock cycles per instructions.

CPI (clock cycles per instructions): CPI is the average number of clock cycles each instruction takes to execute.

CPU Performance Equation:

CPU time = Instruction count * CPI * Clock cycle time

$$= \frac{\text{Instruction count * CPI}}{\text{Clock rate}}$$

Exercises

Practice Problems I

Directions for questions 1 to 20: Select the correct alternative from the given choices.

1. An instruction is stored at location 301 with its address field at location 300. The address field has the value 400. A processor register R_1 contains the number 200. Evaluate the effective address and Match the following:

(A) Direct	(1) 702
(B) Immediate	(2) 600
(C) Relative	(3) 301
(D) Register Indirect	(4) 400
(E) Index with R_1 as index Register	(5) 200

(A) A – 4, B – 3, C – 1, D – 5, E – 2
(B) A – 3, B – 4, C – 1, D – 5, E – 2
(C) A – 4, B – 3, C – 1, D – 2, E – 5
(D) A – 3, B – 3, C – 1, D – 2, E – 5

2. The two word instruction is stored in memory at an address designated by symbol W. The address field of the instruction (stored at $W + 1$) is designated by the symbol Y. The operand used during the execution of the instruction is stored at address symbolized by Z. An index register contains the value X. State how Z is calculated from the other addresses if the addressing mode of the instruction is

(A) Direct	(1) $Z = \text{Mem}(Y)$
(B) Indirect	(2) $Z = Y + W + 2$

(C) Relative (3) $Z = Y + X$
(D) Indexed (4) $Z = Y$

(A) A – 4, B – 1, C – 2, D – 3
(B) A – 3, B – 1, C – 2, D – 4
(C) A – 4, B – 2, C – 1, D – 3
(D) A – 3, B – 2, C – 1, D – 4

3. A computer has 32-bit instruction and 12-bit addresses. If there are 250 two-address instructions, how many one-address instructions can be formulated?
(A) 6 (B) 256
(C) 12,288 (D) 24,576

4. The memory unit of a computer has 256k words of 32-bits each. The computer has an instruction format with four fields:

1. An operation field
2. A mode field to specify one of 8 addressing modes
3. A Register address field to specify one of 120 processor registers.
4. A memory address.

Then what is the number of bits in each field respectively if the instruction is in one memory word?
(A) 4, 3, 7, 18 (B) 3, 4, 7, 18
(C) 2, 1, 6, 23 (D) 3, 7, 4, 18

Common data for questions 5 to 7: A relative mode branch type of instruction is stored in memory at an address equivalent to decimal 750. The branch is made to an address equivalent to decimal 400.

5. What should be the value of the relative address field of the instruction in decimal?
(A) 351 (B) –351
(C) 350 (D) –350

6. The relative address value in binary using 12-bits, will be
(A) 000101011111 (B) 100101011111
(C) 111010100000 (D) 111010100001

7. What will be the binary value in PC after the fetch phase (in binary)?
(A) 001011101111 (B) 001011101110
(C) 111011101111 (D) 110100010001

Common data for questions 8 to 10: Consider a 16-bit processor in which the following appears in main memory, starting at location 200:

200	Load to AC	Mode
201	500	
202	Next to instruction	

The first part of the first word indicates that this instruction loads a value into an accumulator. The mode field specifies an addressing mode or a source register, R_1, which has a value 400. There is a base register that contain the value 100. The value 500 in location 201, may be the part of address calculation. Assume that location 399 contains the value 999, location 400 contains the value 1000 and so on.

8. What will be the effective address and operand to be loaded by using Register indirect mode?
(A) 200, 400 (B) 400, 1000
(C) 400, 500 (D) 200, 1000

9. What will be the effective address using indirect addressing mode?
(A) 200 (B) 201
(C) 500 (D) Present in 500 location

10. What will be the effective address using immediate addressing mode?
(A) 202 (B) 201
(C) 500 (D) 400

11. A CPU of a computer has 48-bit instructions. A program starts at address $(600)_{10}$. Which one of the following is a legal program counter value in decimal?
(A) 610 (B) 650
(C) 672 (D) 693

12. Consider a new instruction named branch- on-bit-reset (bbr). The instruction '*BBR* R_1, *I*, label'

Jumps to label, and if bit in position I of register operand, R_1 is zero. The registers of the computer are 16-bits wide and are numbered 0 to 15, position 0 being LSB. Consider the following implementation of this instruction on a processor that does not have *BBR* implemented.

Temp ← R_1 and mask

Branch to label if temp is zero.

The variable 'temp' is a temporary register. For correct implementation, the variable 'mask' must be generated by
(A) mask ← $0 \times 1 << I$
(B) mask ← $0 \times FFFFFFFF >> I$
(C) mask ← I
(D) mask ← $0 \times F$.

13. Consider a hypothetical processor with an instruction of type

LW R_1, $40(R_2)(R_3)$.

Which during execution reads a 16-bit word from memory and stores it in a 16-bit register R_1. The effective address of the memory location is obtained by the addition of constant 40, contents of R_2 and R_3 registers. Which of the following best reflects the addressing mode implemented by this instruction for the operand in memory?
(A) Immediate addressing
(B) Register addressing
(C) Register indirect scaled addressing
(D) Base with index and displacement addressing

14. Which of the following is true of base-register addressing mode?
(i) It is useful in creating self-relocating code.
(ii) If it is included in an instruction set architecture, then an additional ALU is required for effective address calculation.
(iii) The amount of displacement depends on the content of base register.
(A) (i) only (B) (ii) only
(C) (i) and (ii) only (D) (ii) and (iii) only

15. Which of the following addressing modes are suitable for program relocation at run time?
(i) Direct addressing
(ii) Based register addressing
(iii) PC-relative addressing
(iv) Index register addressing
(A) (i) and (ii) (B) (ii) and (iii)
(C) (iii) and (iv) (D) (ii), (iii) and (iv)

16. In which of the following addressing mode, the address of the operand is inside the instruction?
(A) Implied mode
(B) Absolute addressing mode
(C) Immediate addressing mode
(D) Register addressing mode

17. A certain processor supports only the immediate and the direct addressing modes. Which of the following programming language features can be on this processor?
(i) Pointers
(ii) Arrays
(iii) Initialization
(A) (i) and (ii) (B) (i) and (iii)
(C) (ii) and (iii) (D) (iii) only

18. In which of the following situation, relative addressing mode is useful?
(A) Coroutine writing
(B) Position-independent code writing
(C) Sharable code writing
(D) Interrupt handlers

19. In indexed addressing mode with scaling, the effective address is calculated as
(A) Index + scaling + signed displacement
(B) (Index * scaling) + signed displacement
(C) Index + (scaling * displacement)
(D) (Index + scaling) * displacement

20. Which of the following addressing modes require more number of memory accesses?
(A) DIRECT
(B) IMMEDIATE
(C) INDIRECT
(D) IMPLIED

Practice Problems 2

Directions for questions 1 to 20: Select the correct alternative from the given choices.

1. The addressing mode that facilitates access to an operand whose location is defined relative to the beginning of the data structure in which it appears is
(A) Direct (B) Indirect
(C) Immediate (D) Index

2. Stack addressing is same as
(A) Direct addressing
(B) Indirect addressing
(C) Zero addressing
(D) Relative addressing

3. The Register which contain the Instruction to be executed is called
(A) Instruction register
(B) Memory address register
(C) Index register
(D) Memory data register

4. The Register which keeps track of the execution of a program and which contains the memory address of the next instruction to be executed is called
(A) Instruction register
(B) Program counter
(C) Index register
(D) Memory address register

5. A stack pointer is
(A) A 16-bit register in the microprocessor that indicate the beginning of the Stack Memory
(B) A register that decodes and execute 16-bit arithmetic operation.
(C) The first memory location where a subroutine address is stored
(D) A register in which flag bits are stored.

6. Function of Control Unit in the CPU is
(A) To transfer data to primary storage
(B) To store program instruction
(C) To perform logic operations
(D) To generate timing signals

7. When a subroutine is called the address of the instruction following the CALL instruction stored in the
(A) Stack (B) Accumulator
(C) Program counter (D) Stack pointer

8. In Immediate addressing mode the operand is placed
(A) In the CPU Register
(B) After the OP Code in the instruction
(C) In the memory
(D) In the stack memory

9. When the RET instruction at the end of subroutine is executed
(A) The information where the stack is initialized is transferred to the stack pointer.
(B) The memory address of the RET instruction is transferred to the program counter.
(C) Two data bytes stored in the top two locations of the stack are transferred to the program counter.
(D) Two data bytes stored in the top two location of the stack are transferred to the stack pointer.

10. Match the following:

List I	List II
P. Indirect Addressing	**1.** Loops
Q. Auto decrement Addressing	**2.** Constants
R. Immediate Addressing	**3.** Pointers

(A) P – 1, Q – 3, R – 2
(B) P – 3, Q – 1, R – 2
(C) P – 2, Q – 1, R – 3
(D) P – 3, Q – 2, R – 1

11. An instruction used to set the carry flag in a computer can be
(A) Data control (B) Process control
(C) Logical (D) Data transfer

12. The addressing mode in which the address of the location of the operand is given explicitly as part of the instruction is
(A) Direct addressing mode
(B) Indirect addressing mode
(C) Immediate addressing mode
(D) Register addressing mode

13. The unit that is used to supervise each instructions in the CPU is
(A) Control register (B) Control logic unit
(C) ALU (D) Address register

14. The address of the location to or from which data are to be transferred is called
(A) Memory data register
(B) Memory address register
(C) Program counter
(D) Index register

15. Which register is used as a working area in CPU?
(A) Program counter (B) Accumulator
(C) Stack pointer (D) Instruction register

16. Which of the following statement is false about the PC relative addressing mode?
(A) It allows indexing of array element with same instruction.
(B) It enables reduced instruction size.
(C) It enables faster address calculations than indirect addressing.
(D) It enables easy relocation of data.

17. Which of the following is not an application of logic operations?
(A) Insert new bit values into a register
(B) Change bit value
(C) Delete a group of bits
(D) Shift bit values in a register

18. In which of the following addressing mode, less number of memory references are required?
(A) Immediate (B) Register
(C) Implied (D) All of the above

19. Which of the following is not involved in a memory write operation?
(A) MDR (B) MAR
(C) PC (D) Data bus

20. In ____ addressing mode the instruction contains 8-bit signed offset, address register A_n and index register R_K.
(A) Basic index (B) Full index
(C) Basic relative (D) Full relative

Previous Years' Questions

Common Data for Questions 1 to 3: Consider the following program segment. Here R_1, R_2 and R_3 are the general purpose registers.

	Instruction	Operation	Instruction Size (No. of Words)
	MOV R_1, (3000)	$R_1 \leftarrow$ M[3000]	2
LOOP:	MOV R_2, (R_3)	$R_2 \leftarrow$ M[R_3]	1
	ADD R_2, R_1	$R_2 \leftarrow R_1 + R_2$	1
	MOV (R_3), R_2	M[R_3] $\leftarrow R_2$	1
	INC R_3	$R_3 \leftarrow R_3 + 1$	1
	DEC R_1	$R_1 \leftarrow R_1 - 1$	1
	BNZ LOOP	Branch on not zero	2
	HALT	Stop	1

Assume that the content of memory location 3000 is 10 and the content of the register R_3 is 2000. The content of each of the memory locations from 2000 to 2010 is 100. The program is loaded from the memory location 1000. All the numbers are in decimal.

1. Assume that the memory is word addressable. The number of memory references for accessing the data

in executing the program completely is: **[2007]**
(A) 10 (B) 11
(C) 20 (D) 21

2. Assume that the memory is word addressable. After the execution of this program, the content of memory location 2010 is: **[2007]**
(A) 100 (B) 101
(C) 102 (D) 110

3. Assume that the memory is byte addressable and the word size is 32 bits. If an interrupt occurs during the execution of the instruction 'INC R_3', what return address will be pushed on to the stack? **[2007]**
(A) 1005 (B) 1020
(C) 1024 (D) 1040

4. Which of the following is/are true of the auto-increment addressing mode?
(i) It is useful in creating self-relocating code
(ii) If it is included in an Instruction Set Architecture, then an additional ALU is required for effective address calculation
(iii) The amount of increment depends on the size of the data item accessed **[2008]**
(A) (i) only (B) (ii) only
(C) (iii) only (D) (ii) and (iii) only

5. Consider a hypothetical processor with an instruction of type LW R_1, 20(R_2), which during execution reads a 32-bit word from memory and stores it in a 32-bit register R_1. The effective address of the memory location is obtained by the addition of a constant 20 and the contents of register R_2. Which of the following best reflects the addressing mode implemented by this instruction for the operand in memory? **[2011]**
(A) Immediate addressing
(B) Register addressing
(C) Register indirect scaled addressing
(D) Base indexed addressing

6. Consider two processors P_1 and P_2 executing the same instruction set. Assume that under identical conditions, for the same input, a program running on P_2 takes 25% less time but incurs 20% more CPI (Clock cycles per instructions) as compared to the program running on P_1. If the clock frequency of P_1 is 1GHz, then the clock frequency of P_2 (in GHz) is _____. **[2014]**

7. A machine has a 32-bit architecture with 1-word long instructions. It has 64 registers, each of which is 32 bits long. It needs to support 45 instructions, which have an immediate operand in addition to two register operands. Assuming that the immediate operand is an unsigned integer, the maximum value of the immediate operand is _____. **[2014]**

8. Consider a new instruction named branch-on-bit-set (mnemonic bbs). The instruction 'bbs reg, pos, label' jumps to label if bit in position pos of register operand reg is one. A register is 32 bits wide and the bits are numbered 0 to 31, bit in position 0 being the least significant. Consider the following emulation of this instruction on a processor that does not have bbs implemented.

temp ← reg & mask

Branch to label if temp is non-zero.

The variable temp is a temporary register. For correct emulation, the variable mask must be generated by **[2006]**
(A) mask ← 0 × 1 << pos
(B) mask ← 0 × ffffffff >> pos
(C) mask ← pos
(D) mask ← 0 × f

9. Consider a processor with byte-addressable memory. Assume that all registers, including Program Counter (PC) and Program Status Word (PSW), are of size 2 bytes. A stack in the main memory is implemented from memory location $(0100)_{16}$ and it grows upward. The stack pointer (SP) points to the top element of the stack. The current value of SP is $(016E)_{16}$. The CALL instruction is of two words, the first word is the op-code and the second word is the starting address of the subroutine (one word = 2 bytes). The CALL instruction is implemented as follows:

- Store the current value of PC in the stack
- Store the value of PSW register in the stack
- Load the starting address of the subroutine in PC

The content of PC just before the fetch of a CALL instruction is $(5FA0)_{16}$. After execution of the CALL instruction, the value of the stack pointer is **[2015]**
(A) $(016A)_{16}$ (B) $(016C)_{16}$
(C) $(0170)_{16}$ (D) $(0172)_{16}$

10. A processor has 40 distinct instructions and 24 general purpose registers. A 32 - bit instruction word has an opcode, two register operands and an immediate operand. The number of bits available for the immediate operand field is ____. **[2016]**

11. Suppose the functions F and G can be computed in 5 and 3 nanoseconds by functional units U_F and U_G, respectively. Given two instances of U_F and two instances of U_G, it is required to implement the computation $F\ (G(X_i))$ for $1 \le i \le 10$. Ignoring all other delays, the minimum time required to complete this computation is _____ nanoseconds. **[2016]**

12. Consider a processor with 64 registers and an instruction set of size twelve. Each instruction has five distinct fields, namely, opcode, two source register identifiers, one destination register identifier, and a twelve - bit immediate value. Each instruction must be stored in

memory in a byte - aligned fashion. If a program has 100 instructions, the amount of memory (in bytes) consumed by the program text is ___. **[2016]**

13. Consider the C struct defined below:

```
struct data  {
    int marks [100];
    char grade;
    int cnumber;
};
struct data student;
```

The `base` address of `student` is available in register R1. The field `student.grade` can be accessed efficiently using **[2017]**

(A) Post-increment addressing mode, $(R1)+$
(B) Pre-decrement addressing mode, $-(R1)$
(C) Register direct addressing mode, $R1$
(D) Index addressing mode, $X(R1)$, where X is an offset represented in 2's complement 16-bit representation.

14. Consider a RISC machine where each instruction is exactly 4 bytes long. Conditional and unconditional branch instructions use PC-relative addressing mode Offset specified in bytes to the target location of the branch instruction. Further the Offset is always with respect to the address of the next instruction in the program sequence. Consider the following instruction sequence.

Instr. No.	Instruction
i :	add R2, R3, R4
i+1 :	sub R5, R6, R7
i + 2 :	cmp R1, R9, R10
i + 3 :	beq R1, Offset

If the target of the branch instruction is i, then the decimal value of the Offset is _____. **[2017]**

15. A processor has 16 integer registers ($R0, R1, ..., R15$) and 64 floating point registers ($F0, F1, ..., F63$). It uses a 2-byte instruction format. There are four categories of instructions: Type-1, Type-2, Type-3, and Type-4. Type-1 category consists of four instructions, each with 3 integer register operands ($3Rs$). Type-2 category consists of eight instructions, each with 2 floating point register operands ($2Fs$). Type-3 category consists of fourteen instructions, each with one integer register operand and one floating point register operand ($1R + 1F$). Type-4 category consists of N instructions, each with a floating point register operand ($1F$).

The maximum value of N is _____. **[2018]**

Answer Keys

Exercises

Practice Problems 1

1. A	**2.** A	**3.** D	**4.** A	**5.** B	**6.** D	**7.** A	**8.** B	**9.** D	**10.** B
11. C	**12.** A	**13.** D	**14.** A	**15.** B	**16.** B	**17.** B	**18.** B	**19.** B	**20.** C

Practice Problems 2

1. D	**2.** C	**3.** A	**4.** B	**5.** A	**6.** D	**7.** A	**8.** B	**9.** C	**10.** B
11. B	**12.** A	**13.** B	**14.** B	**15.** B	**16.** A	**17.** D	**18.** B	**19.** D	**20.** B

Previous Years' Questions

1. D	**2.** A	**3.** C	**4.** C	**5.** D	**6.** 1.6	**7.** 16383	**8.** A	**9.** D	**10.** 16
11. 28	**12.** 500	**13.** D	**14.** –16	**15.** 32					

Chapter 2

ALU and Data Path, CPU Control Design

LEARNING OBJECTIVES

- *Arithmetic and logic unit*
- *Fixed-point arithmetic operation*
- *Floating point arithmetic operation*
- *BCD*
- *Data path*
- *CPU control design*
- *Instruction cycle*
- *Control unit*
- *Control of processor*
- *Function of control unit*
- *Design of control unit*
- *Types of micro-instructions*
- *Micro-instruction sequencing*
- *RISC and CISC*
- *RISC characteristic*
- *CISC characteristic*

ALU (Arithmetic and Logic Unit)

ALU performs arithmetic and logical operations on data (*see* Figure 1).

Figure 1 ALU inputs and outputs

- Data are presented to ALU in registers and the results of an operation are stored in registers.
- Registers are temporary storage locations within the processor that are connected by signal paths to ALU.
- The control unit provides signals that control the operation of ALU and the movement of data into and out of the ALU.
- Here we will discuss
  1. Fixed-point arithmetic operations
  2. Floating-point arithmetic operations
  3. BCD data arithmetic operations

Fixed-point Arithmetic Operations

Fixed-point representation

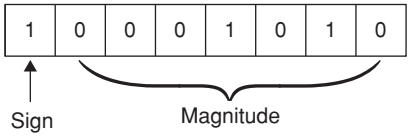
The numbers may be positive, zero or negative. So we have two types of numbers:

Unsigned numbers Only zero and positive integers can be represented. All bits represent magnitude and no need of sign.

Signed numbers In signed representation, the most significant bit represents the sign. If the number is positive, the MSB is 0 and remaining bits represent magnitude. If the number is negative, we have three techniques to represents that number:

1. **Signed magnitude representation:** In signed magnitude representation, the MSB represents sign and remaining bits represents magnitude. If the number is negative then the MSB is 1.

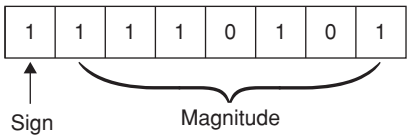
   Example: Signed magnitude representation of –10 =



2. **Signed 1's complement representation:** In signed 1's complement representation, the MSB bit is 1. The remaining bits of its signed magnitude bits are inverted i.e., convert 0's to 1's and 1's to 0's to obtain 1's complement.

Example:

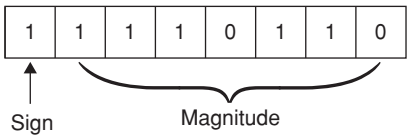
Signed 1's complement (–10) =



3. **Signed 2's complement representation:** To get signed 2's complement representation, add 1 to the signed 1's complement of that number.

Example:

Signed 2's complement (–10) =



Fixed-point arithmetic operations

We will discuss the following operations using signed magnitude data and signed 2's complement data.

1. Addition
2. Subtraction
3. Multiplication
4. Division

Addition and subtraction using signed magnitude data Consider two numbers whose magnitude is represented as A and B. When the signed numbers are added or subtracted, there are eight different conditions to consider, depending on the sign of the numbers and operation performed.

Operation	Add Magnitudes	Subtract Magnitudes ($A > B$)
$(+A) + (+B)$	$+(A + B)$	
$(+A) + (-B)$		$+(A - B)$
$(-A) + (+B)$		$-(A - B)$
$(-A) + (-B)$	$-(A + B)$	
$(+A) - (+B)$		$+(A - B)$
$(+A) - (-B)$	$+(A + B)$	
$(-A) - (+B)$	$-(A + B)$	
$(-A) - (-B)$		$-(A - B)$

Algorithm for addition (subtraction): When the signs of A and B identical (different), add the two magnitudes and attach the sign of A to the result. When the signs of A and B are different (identical), compare the magnitudes and subtract the smaller number from the larger. Choose the sign of result based on magnitudes of A and B.

Example: All eight cases for the numbers $A = 5$, $B = 2$.

$(+A) + (+B) = (+5) + (+2)$

$= 0101 + 0010 = 0111 = +7$

$(+A) + (-B) = (+5) + (-2)$

$= 0101 + 1010$

Take 2's complement of –2 and add it to 5

101

110

1] 011

↑

Discard

$\therefore$ result = + 3 ($A > B$)

$(-A) + (+B) = (-5) + (+2)$

$= 1101 + 0010$

add 2's complement of –5 to 2

011

010

101

As MSB is 1 take 2's complement to get original number i.e., 011.

Result = 1011 = –3 ($\because A > B$)

$(-A) + (-B) = (-5) + (-2)$

$= 1101 + 1010$

101

010

Result = 1111 = –7

Similarly we can perform the subtractions using signed magnitude data.

Hardware implementation:

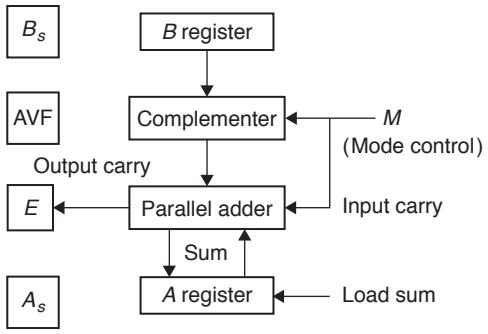


Figure 2 Hardware implementation for addition and subtraction.

Figure 2 shows the hardware implementation for addition and subtraction operations. It consists of registers A and B and sign flip-flops A_s and B_s. Subtraction is done by adding A to the 2's complement of B. The output carry is transferred to flip-flop E and add overflow flip-flop AVF holds the overflow bit when A and B are added. The addition is done through the parallel adder. The output of adder is sent to 'A' register. The complementer provides an output of B or complement of B depending on the state of mode control M. When $M = 0$, the output equal to $A + B$, when $M = 1$, the output equal to $A + \bar{B} + 1$, i.e., $A - B$.

Addition and Subtraction with signed 2's complement data

Addition: In 2's complement representation, addition proceeds as if the two numbers were unsigned integers. If the result of the operation is positive, we get a positive number

in 2's complement form, which is same as in unsigned integer form. If the result of the operation is negative, we get a negative number in 2's complement form.

Example:

+5 = 0101
+2 = 0010
0111 = +7

+5: 0101
–2: 1110
10011 = +3
–5: 101
+2: 0010
1101

As the result is negative, take 2's complement of result to get original number, i.e., 0011 and the answer is –3.

–5: 1011
–2: 1110
11001

As the result is negative take 2's complement to get original number, i.e., 0110 + 1 = 0111.

∴ Answer is –7.

Note: If two numbers are added and they are both positive or both negative, then overflow occurs if the result has the opposite sign.

Subtraction: To subtract subtrahend from minuend, take the 2's complement of subtrahend and add it to the minuend.

Example:

+5: 0101
+2: 0010

To subtract these two numbers add 2's complement of 2 to 5.

+5: 0101
–2: 1110
10011 = +3
+5: 0101
– 2: 1110
2's complement of –2 = 0010.
+5: 0101
+2: 0010
0111 = +7

Similarly for the other cases we can perform the subtraction.

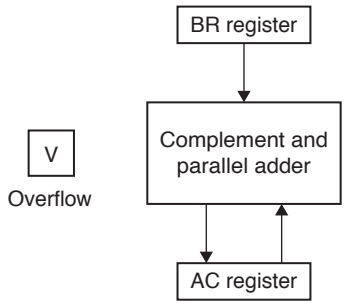


Figure 3 Hardware implementation for signed 2's complement addition and subtraction:

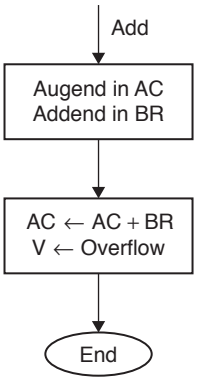


Figure 4 Flowchart for addition in 2's complement form

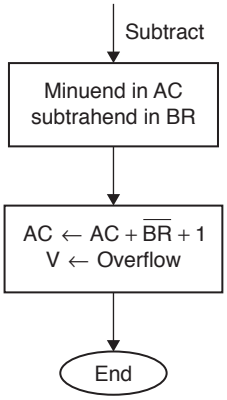


Figure 5 Flowchart for subtraction of 2's complement data

Multiplication of signed magnitude data

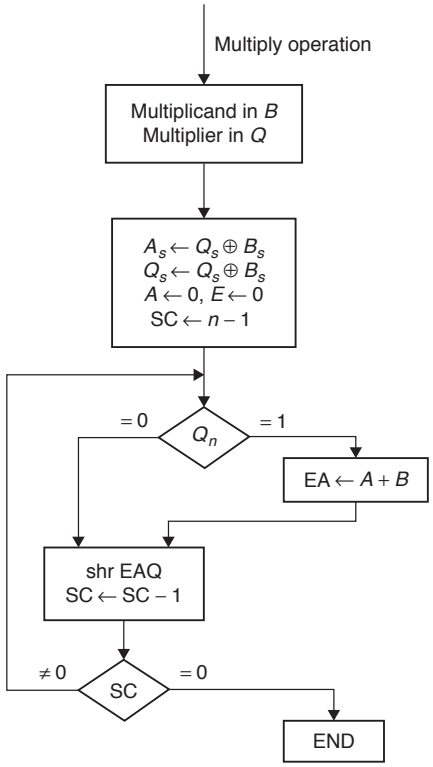


Figure 6 Flowchart for multiplication of signed magnitude data

Multiplication of two fixed point binary numbers in signed magnitude representation is a process of successive shift and add operations (*see* Figure 6).

Example 1: Multiply the two numbers −7 and +8, using 5-bit registers.
−7 = 10111
+8 = 01000

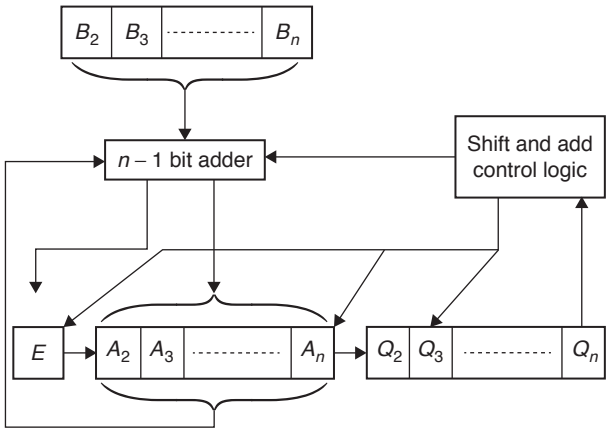
By excluding sign-bits, the multiplicand, $B = 0111$ and multiplier $Q = 1000$. Initially $A = 0000$, SC is sequence counter contains number of bits in multiplier magnitude.

Here SC = 4

Multiplicand B = 0111	***E***	***A***	***Q***	**SC**
Multiplier in *Q*	0	0000	1000	4
Last bit of *Q*, $Q_n = 0 \Rightarrow$ Shr *EAQ*	0	0000	0100	3
$Q_n = 0 \Rightarrow$ Shr *EAQ*	0	0000	0010	2
$Q_n = 0 \Rightarrow$ Shr *EAQ*	0	0000	0001	1
$Q_n = 1 \Rightarrow$ Add *B* to *A*	0	0000 0111 0111	0001	
Shr *EAQ*	0	0011	1000	0

$B \times Q = 00111000 = 56$
Sign = $Q_s \oplus B_s = 1 \oplus 0 = 1$
∴ Result = –56

Hardware for signed magnitude data multiplication: B_1, A_1, Q_1 represent the respective signs of the registers B, A, Q. Final result will be in AQ, which consist of $2n$-bits. (Here each register has n-bits).



Multiplication of Singed 2's complement data The straight forward multiplication will not work if either the multiplicand or the multiplier is negative. There are number of ways to perform multiplication of signed 2's complement data. One such a technique is *Booth's multiplication algorithm*. The following flowchart depicts about Booth's algorithm (see figure 7).

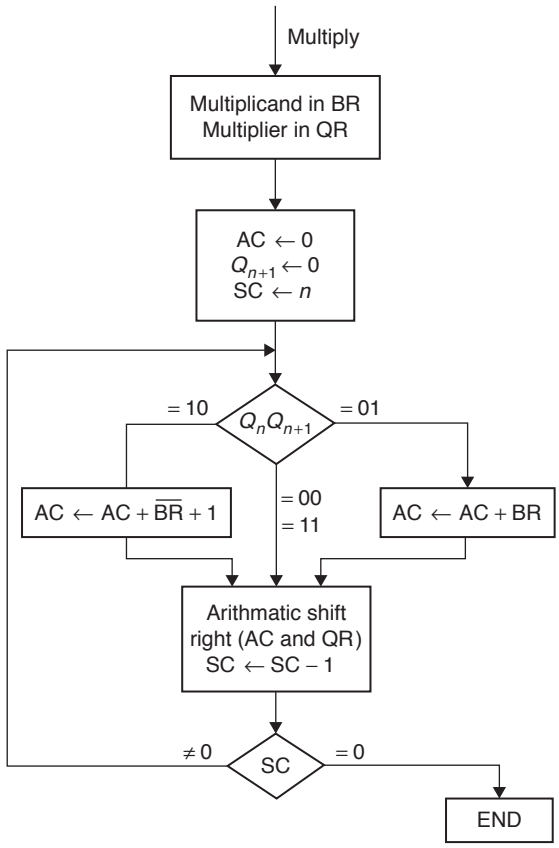


Figure 7 Booth's multiplication algorithm

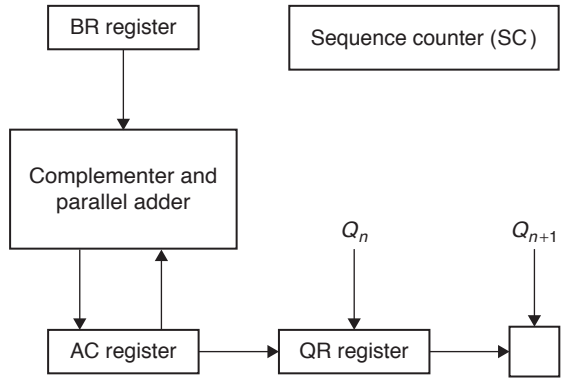
An additional 1-bit register placed logically to the right of the LSB(Q_n) of Q register designated Q_{n+1}.

Example 2: Multiply the two numbers −7 and +8 using booth's algorithm, using 5 bits.
BR = −7 = 11001
QR = +8 = 01000
Initially AC = 00000, $Q_{n+1} = 0$, SC = 5

$Q_n\, Q_{n+1}$	**BR = 11001** $\overline{\text{BR}} + 1 = 00111$	**AC**	**QR**	Q_{n+1}	**SC**
	Initial	00000	01000	0	5
00	ashr(AC and QR)	00000	00100	0	4
00	ashr(AC and QR)	00000	00010	0	3
00	ashr(AC and QR)	00000	00001	0	2
10	subtract BR ashr(AC and QR)	00000 00111 00111 00011			
			10000	1	1
01	add BR ashr(AC and QR)	00011 11001 11100 11110	 10000 01000	 1 0	 1 0

∴ Result = 1111001000 = –56

Hardware implementation for Booth's algorithm:



Note: These two multiplication algorithms are sequential but we can also do the operation by means of a combinational circuit that forms the product bits all at once. The circuit consist of AND gates and adders.

Division algorithms

Division of signed magnitude data: Division of signed magnitude data is a process of successive compare, shift and subtract operations.

Example:

Dividend = 0111000000
Divisor = 10001

```
10001) 0111000000 (11010
       –10001
        010110
        –10001
         0010100
          –10001
           000110
```

Hardware implementation:

- The hardware implementation of division is same as multiplication, instead of shifting the divisor to the right, the dividend or partial remainder is shifted to the left, thus leaving the two numbers in the required relative position.
- The divisor is stored in the B register and the double-length dividend is stored in registers A and Q. The dividend is shifted to the left and the divisor is subtracted by adding its 2's complement value. The information about the relative magnitude is available in E.
- If $E = 1$, it signifies that $A \geq B$. A quotient bit 1 is inserted into Q_n and the partial remainder is shifted to the left to repeat the process.
- If $E = 0$, it signifies that $A < B$ so the quotient is Q_n remains a 0. The value of B is added to restore the partial remainder in A to its previous value. The partial remainder is shifted to the left and the process is repeated again until all quotient bits are formed.
- Finally, the quotient is in Q and remainder is in A. This method is called *restoring method*.

Divide overflow:

- A divide overflow condition occurs if the high-order half bits of the dividend constitute a number greater than or equal to the divisor.
- A division by zero must be avoided.

Other algorithms for division: Two other methods are available for dividing numbers:

Comparison method: To divide the two numbers A and B in comparison method, they are compared prior to the subtraction operation. If $A \geq B$, B is subtracted from A. If $A < B$ nothing is done. The partial remainder is shifted left and the numbers are compared again.

Non-restoring method: To divide two numbers A and B is non-restoring method, B is not added if the difference is negative but instead, the negative difference is shifted left and then B is added.

Floating-point Arithmetic Operations

Floating-point representation

Fixed-point representation allows representation of numbers with fractional component as well. But this approach has limitations. It is not possible to represent very large numbers and very small numbers in fixed point representation.

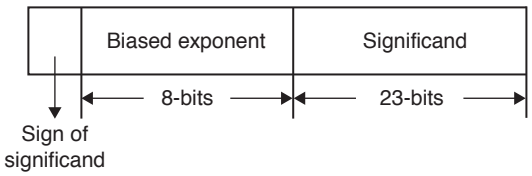
In floating-point representation, the numbers can be represented in the form,

$\pm S \times B^{\pm E}$

The three fields are
Sign: plus or minus
Significand: S
Exponent: E
are stored in a binary word.

The base B is implicit and need not be stored because it is same for all the numbers. It is assumed that the radix point is to the right of the left most or most significant bit of the significand, i.e., there is one bit to the left of the radix point.

32-bit floating-point format: The left most bit stores the sign of the number. The exponent value is stored in next 8-bits. This is represented in biased representation.



Biased representation: In biased representation, a fixed value, called the bias, is subtracted from the exponent field to get the true exponent value.

Bias = $(2^{k-1} - 1)$, where k = number of bits in binary exponent. In IEEE 32-bit floating point representation, bias = $2^7 - 1 = 127$.

And the range of true exponents is −127 to +128.

The advantage of biased representation is that non-negative floating point numbers can be treated as integers for comparison purposes.

The last portion of word is the siginificand.

Normalized numbers:

- To simplify operation on floating point numbers, it is typically required that they must be normalized.
- A normalized number is one in which the most significant digit of significand is non-zero.
- For base-2 representation, a normalized number is therefore one in which the MSB of the significand is one.
- Normalized non-zero number is one in the form $\pm 1.bbb\cdots b \times 2^{\pm E}$, where b is either binary digit 0 or 1.
- As the MSB is always one, it is unnecessary to store this bit. Thus the 23-bit field is used to store a 24-bit significand with a value in the half open internal (1, 2).
- A number may be normalized by shifting the radix point to the right of the leftmost 1 bit and adjusting the exponent accordingly.

Example: In 32-bit floating representation of $-1.6328125 \times 2^{-20}$,

Sign = 1 (as the number is negative)

$(.6328125)_{10} = (.1010001\ldots)_2$

Exponent = −20

Biased exponent = 127 − 20 = 107

= 1101011

$\therefore -1.6328125 \times 2^{-20}$

= 1 01101011 10100010000000000000000

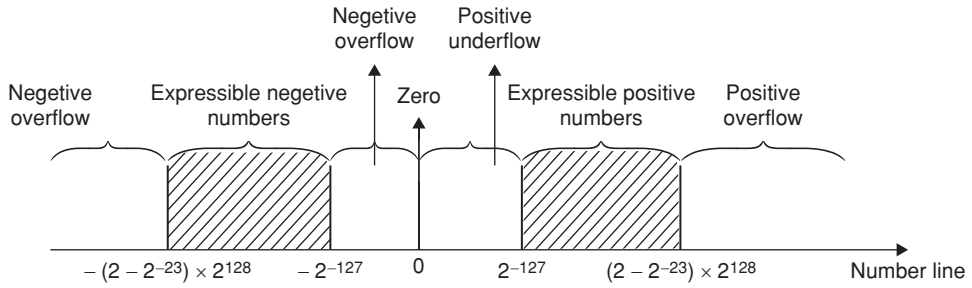
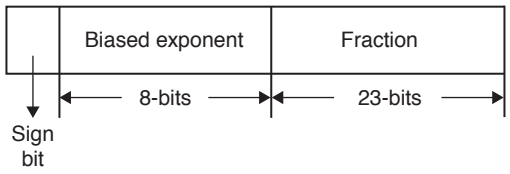
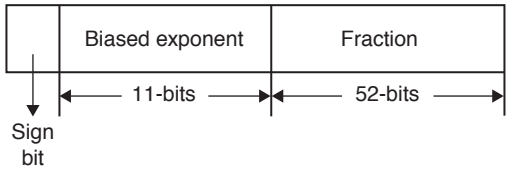


Figure 8 Range of expressible numbers in a 32-bit floating point format

IEEE standard for binary floating-point representation



Single precision format



Double precision Format

Floating-point arithmetic

(i) Addition and subtraction: The algorithm consists the following phases:

1. Check for zeros
2. Align the significands/mantissas
3. Add or subtract the significands
4. Normalize the result

Example:

$(123 \times 10^0) + (234 \times 10^{-2})$

$=123 \times 10^0 + 2.34 \times 10^0 = 125.34 \times 10^0$

(ii) Multiplication: The steps to multiply two floating point numbers are

1. Check for zeros
2. Add the exponents
3. Multiply the significands
4. Normalize the result

(iii) Division: The steps to divide two floating point numbers are

1. Check for zeros
2. Initialize registers and evaluate the sign
3. Align the dividend
4. Subtract the exponents
5. Divide the significands

Binary-Coded Decimal (BCD) Arithmetic Operations

Computers capable of performing decimal arithmetic must store the data in binary-coded form.

Example: BCD of 239 = 0010 0011 1001

BCD addition

In BCD each digit do not exceed 9, so the sum of two BCD digits cannot be greater than 9 + 9 + 1 = 19, the 1 in the sum being an input carry. When the binary sum of two BCD digits is greater than 1001, we obtain a non-valid BCD representation. The addition of binary 6 (0110) to the binary sum converts it to the correct BCD representation and also produces an output carry as required.

Example:

239 = 0010 0011 1001
426 = 0100 0010 0110
665 0110 0101 1111 >1001
0110
0110 0110 0101
= 665

BCD subtraction

- Perform the subtraction by taking the 9's or 10's complement of the subtrahend and adding it to the minuend.
- The 9's complement of a decimal digit represented in BCD can be obtained by complementing the bits in the coded representation of the digit, provided a correction is included.

There are two possible correction methods:

1. Binary 1010 is added to each complemented digit and the carry discarded after each addition.

Example:

9's complement of 7 = 2
7 in BCD = 0111.
Complement of 7 = 1000
Add 1010 = 1010
1] 0010 = 2
↑
Discard

2. Binary 0110 is added before the digit is complemented.

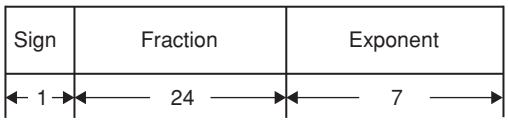
Example:

BCD of 7 = 0111
Add 0110 = 0110
1101
Complement = 0010 = 2

Example 4: Which of the following multiplier bit pattern of Booth's multiplication algorithm gives worst case performance?
(A) 01010101….0101
(B) 000000….0000
(C) 11111111….1111
(D) 011110111110….01110

Solution: (A)
Booth's multiplication algorithm works well with consecutive 0's or 1's. But it gives worst case performance when the multiplier consists of alternative 0's and 1's. (As 01, 10 pattern leads to addition and subtractions).

Example 5: Consider the following 32-bit floating point representation scheme as shown in the format below:



A value is specified by three fields:
Sign field: 1 bit (0 for positive and 1 for negative values)
Fraction: 24-bits (with binary point being at the left end of fraction bits)
Exponent:7-bits (in excess-64 signed integer representation)
The base of exponentiation is 16. The sign bit is in MSB. Then the normalized floating point representation of – 6.5 is

(A) E8000042 (B) E1000012
(C) D8000841 (D) D0000042

Solution: (D)

Here sign = 1 as the number is negative.
$(-6.5)_{10} = (-0110.1)_2$
$= (-1.101 \times 2^2)$
Fraction = 10100000000000000000000
Exponent = Excess – 64 exponent
= 64 + 2 = 66 = 1000010
$\therefore (-6.5)_{10}$
= 1 10100000000000000000001000010
= D0000042.

Data Path

Data path consists of the components of the processor that performs arithmetic operations.

Components of data path: ALU is just one data path building block. Other components are

1. Computational Components, which consist of combinational circuits (output follow inputs)

Example: ALU.

2. State components, which consists of sequential circuits (output changes on clock edge)

Example: Registers.

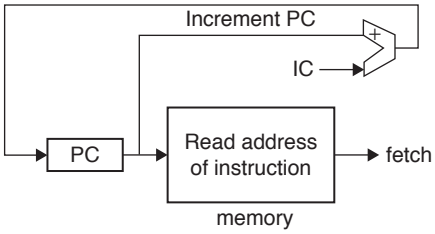
Example: The sequence of steps for the addition of two registers content are

1. $R1_{out}, X_{in}$
2. $R2_{out}$, Choose X, ADDITION, Y_{in}
3. $Y_{out}, R3_{in}$

(Each step executed in a single clock cycle).

- Data path and control unit forms the processing unit of a computer. The Data path includes ALU, multiplexers, all registers (like PC, IR) etc.

Example Data Path Design:

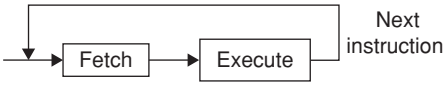


CPU Control Design

Instruction Cycle

A program residing in the memory unit of the computer consists of a sequence of instructions. The program is executed in the computer by going through a cycle for each instruction. Each instruction cycle in turn is subdivided into a sequence of sub cycles. For example, the phases of instruction cycle may be

1. Fetch
2. Decode
3. Read effective address
4. Execute, etc.



Instruction cycle

The cycle will be repeated, till all the instructions are executed.

Each phase is made up of more fundamental operations, called micro-operations.

Example micro-operations: Transfer between registers, simple ALU operation, etc.

Control Unit

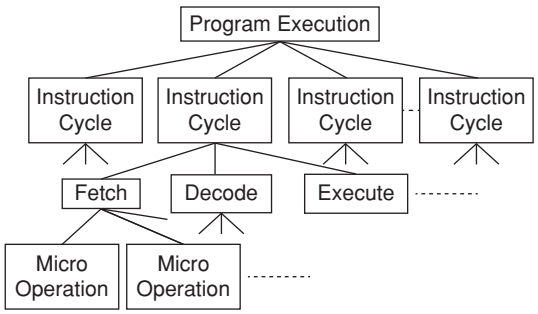
The control unit of a processor performs two tasks:

1. It causes the processor to execute micro-operations in the proper sequence, determined by the program being executed.
2. It generates the control signals that cause each micro-operation to be executed.

We now discuss the micro-operations of various phases of instruction cycle.

Micro-operation

- These are the functional or atomic operations of a processor.



Fetch Cycle: 'Fetch' stage of an instruction occurs at the beginning of each instruction, which causes an instruction to be fetched from memory. The micro-operations involved in fetch phase are

t_1: MAR ← PC (move contents of PC to MAR)
t_2: MBR← memory; PC ← (PC) + I (move contents of MAR location to MBR and increment PC by I)
t_3: IR← (MBR) · (move contents of MBR to IR)

Here *I* is instruction length.
Each micro-operation can be performed within the time of a single time unit.

Execute Cycle: For a machine with *N* different opcodes, there will be *N* different sequence of micro-operations. For the execution of following instruction.
Add R_1, *X*, the micro-operations will be

t_1: MAR← (IR(Address))
t_2: MBR← memory
t_3: R_1← (R_1) + (MBR)

Control of the Processor

(i) Functional requirements of control unit: Let us consider the following concepts to the characterization of a CU.

1. Define the basic elements of the processor.
2. Describe the micro-operations that the processor performs.
3. Determine the functions that the control unit must perform to cause the micro-operations to be performed.

(ii) Basic elements of processor:

- ALU
- Registers
- Internal data path: Used to move data between registers and between register and ALU.
- External data path: Used to link registers to memory and *input–output* modules, often by means of a system bus.
- Control unit: Causes operations to happen within the processor.

(iii) Micro-operations of processor:

- Transfer data from one register to another.
- Transfer data from a register to an external interface.

- Transfer data from an external interface to a register.
- Perform an arithmetic or logic operation, using registers for input and output.

(iv) Control unit tasks:

- **Sequencing:** The control unit causes the processor to step through a series of micro-operations in the proper sequence, based on the program being executed.
- **Execution:** The control unit causes each micro-operation to be executed.

(v) Control signals: For the control unit to perform its function, it must have inputs that allow it to determine the state of the system and outputs that allows it to control the behaviour of the system. These are external specifications of the control unit.

Internally, the control unit must have the logic required to perform its sequencing and execution functions.

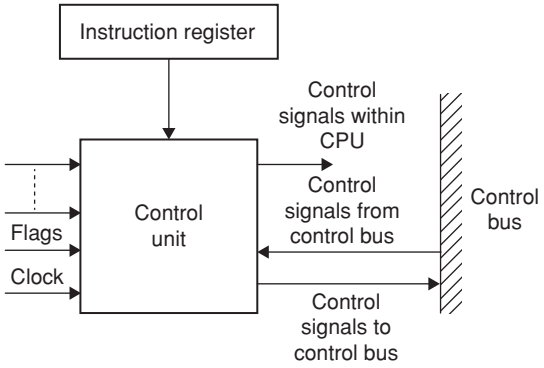


Figure 9 Block diagram of control unit

(a) Clock: This is how the control unit 'keeps time.' The control unit causes one micro-operation to be performed for each clock pulse. This is referred as processor cycle time or clock cycle time.

(b) Instruction registers: The opcode of current instruction is used to determine which micro-operations to perform during the execute cycle.

(c) Flags: Used to determine the status of the processor and outcome of previous ALU operations.

(d) Control signals from control bus: The control bus portion of system bus provides signals to the control unit.

(e) Control signals within the processor:

1. Those that cause data to be moved from register to another.
2. Those that activate specific ALU functions.

(f) Control signals to control bus:

1. Control signals to memory.
2. Control signals to input–output modules.

Totally, there are three types of control signals:

1. Those that activate ALU function.
2. Those that activate a data path.
3. Those that are signals on the external system bus.

Functions of Control Unit

- The control unit directs the entire computer system to carry out stored program instructions.
- The control unit must communicate with both the Arithmetic Logic Unit and Main memory.
- The control unit instructs the arithmetic logic unit by which, logical or arithmetic operation is to be performed.
- The control unit coordinate the activities of the other two units as well as all peripheral and auxiliary storage devices linked to the computer.

Design of Control Unit

Control unit generates control signals using one of the two organizations

(1) Hardwired control unit
(2) Micro-programmed control unit.

Hardwired control unit

- It is implemented as logic circuits (gates, flip-flops, decoders, etc.) in the hardware.
- It is very complicated if we have a large control unit.
- In this organization, if the design has to be modified or changed. It requires changes in wiring among the various components. Thus the modification of all the combinational circuits may be very difficult.

Architecture of hardwired control unit An example hardwired control unit is shown in Figure 9.

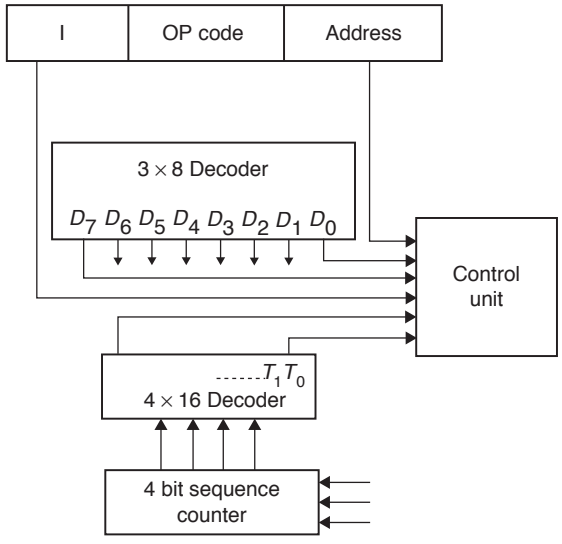


Figure 10 Hardwired control unit

The above control unit consists of:

- Instruction Register
- Number of control logic gates

- Two decoders
- 4-bit sequence counter.
- An instruction read from memory is placed in the instruction register (IR)
- The instruction register is divided into three parts: the *I* bit, operation code and Address part.
- First 12-bits (0-11) to specify an address, next 3-bits specify the operation code (op code) field of the instruction and last left most bit specify the addressing mode *I*.
  $I = 0$ for direct address
  $I = 1$ for indirect address
- First 12-bits are applied to the control logic gates.
- The Opcode bits (12-14) are decoded with 3 × 8 decoder.
- The eight outputs (D_0 through D_7) from a decoder go to the control logic gates to perform specific operation.
- Last bit 15 is transferred to a *I* flip flop designated by symbol *I*.
- The 4-bit sequence counter SC can count in binary from 0 through15.
- The counter output is decoded into 16 timing pulses T_0 through T_{15}.
- The sequence counter can be incremented by INR input or clear by CLR input synchronically.

Advantages:

- Hardwired control unit is fast because control signals are generated by combinational circuits.
- The delay in generation of control signals depends upon the number of gates.

Disadvantages:

- More is the control signal required by CPU, more complex will be the design of control unit.
- Modifications in control signal are very difficult. That means it requires rearranging of wires in the hardware circuit.
- It is difficult to correct mistake in original design or adding new features.

Micro-programming control unit

- A micro-programmed Control unit is implemented using programming approach. A sequence of micro-operations are carried out by executing a program consisting of microinstructions.
- Micro-program, consisting of micro instructions is stored in the control memory of the control unit.
- Execution of micro-instruction is responsible for generation of a set of control signals.

A micro-instruction consists of:

- One or more micro-instructions to be executed.
- Address of next micro-instruction to be executed.

(a) Micro-operations: The operations performed on the Data stored inside the registers are called Micro-operations.

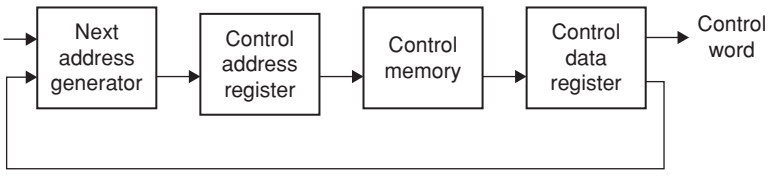
(b) Micro-programs: Micro-programming is the concept for generating control signals using programs. These programs are called Micro-programs.

(c) Micro-instructions: The instructions that make Micro-programs are called micro-instructions.

(d) Micro-code: Micro-program is a group of micro-instructions. Micro-program can also be termed as micro-code.

(e) Control memory: Micro-programs are stored in the read-only memory (ROM). That memory is called control memory.

(f) Architecture of Micro-Programmed Control Unit:



- The address of micro-instruction that is to be executed is stored in the control address register (CAR).
- Micro-instruction corresponding to the address stored in CAR is fetched from control memory and is stored in the control data register (CDR).
- This micro-instruction contains control word to execute one or more micro-operations.
- After the execution of all micro-operations of micro-instructions, the address of next micro-instructions is located.

Advantages:

- The design of micro-program control unit is less complex because micro-programs are implemented using software routines.
- The micro-programmed control unit is more flexible because design modifications, correction and enhancement is easily possible.
- The new or modified instruction set of CPU can be easily implemented by simply rewriting or modifying the contents of control memory.
- The fault can be easily diagnosed in the micro-program control unit using diagnostic tools by maintaining the contents of flags, registers and counters.

Disadvantages:

- The micro-program control unit is slower than hardwired control unit. That means to execute an instruction in micro-program control unit requires more time.

- The micro-program control unit is expensive than hard-wired control unit in case of limited hardware resources.
- The design duration of micro-program control unit is more than hardwired control unit for smaller CPU.

Types of Micro-instructions

Micro-instructions can be classified as

Horizontal micro-instruction

- Individual bits in horizontal micro-instructions correspond to individual control lines.
- These are long and allow maximum parallelism since each bit controls a single control line.
- No decoding needed.

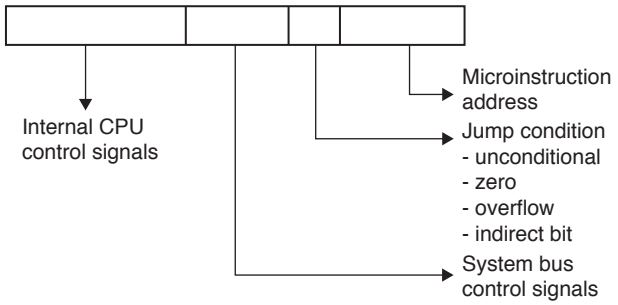


Figure 11 Horizontal micro-instruction format

Vertical micro-instruction

- Here, control lines are coded into specific fields within a micro-instruction.
- Decoders are needed to map a field of k-bits to 2^k possible combinations of control lines.

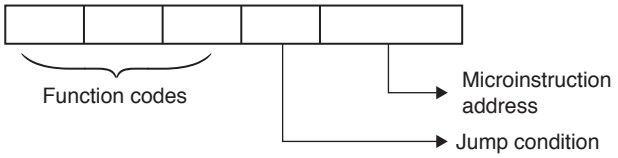


Figure 12 Vertical micro-instruction format

Example: A 3-bit field in a micro-instruction could be used to specify any one of eight possible lines.

- Hence these instructions are much shorter than horizontal ones.
- Control fields encoded in the same field cannot be activated simultaneously. Therefore vertical micro-instructions allow only limited parallelism.
- Decoding is necessary.

Micro-instruction Sequencing

Two concerns are involved in the design of a micro-instruction sequencing technique:

1. The size of micro-instruction: Minimizing size of control memory reduces the cost of that component.
2. The address-generation time:

A desire to execute micro-instructions as fast as possible. In executing a micro program, the address of next micro-instruction to be executed is in one of these categories.

1. Determined by IR
2. Next sequential address
3. Branch

Micro-instructions Execution

The Micro-instruction cycle has two parts:

1. Fetch
2. Execution

The effect of execution of a micro-instruction is to generate control signals. Some of the signals control points internal to the processor. The remaining signals go to the external control bus or other external interface.

Micro-instructions can be classified in a variety of ways.

1. Vertical/horizontal
2. Packed/unpacked
3. Hard/soft micro-programming
4. Direct/indirect encoding.

RISC AND CISC

One of the important aspects of computer architecture is the design of the instruction set for the processor. The instruction set chosen for a particular computer determines the way that machine language programs are constructed. There are two categories of computers based on instructions:

1. Complex instruction set computer (CISC)
2. Reduced instruction set computer (RISC)

CISC: A computer with a large number of instructions is classified as a complex instruction set computer.

RISC: A computer which has fewer instructions with simple constructs, so they can be executed much faster with in the CPU without having to use memory as often. This type of computer is classified as RISC.

CISC characteristics

- CISC provides a single machine instruction for each statement, That is written in a high level language so that compilation process is simplified and the over all computer performance improved.
- It has variable length instruction formats.
- It provides direct manipulation of operands residing in memory.
- Some instructions that perform specialized tasks and are used infrequently.
- A large variety of addressing modes

Drawback of CISC architecture As more instructions and addressing modes are incorporated into a computer, the more hardware logic is needed to implement and support them and hence this causes the computations to slow down.

RISC characteristics

- Reduce execution time by simplifying the instruction set of the computer.
- Fewer numbers of instructions
- Relatively fewer addressing modes
- Memory access is limited to load and store instructions.
- All operations are done with in the register of the CPU.
- Fixed - length, easily decoded instruction format.
- Single-cycle instruction execution.
- Hardwired rather than micro-programmed control.
- Relatively large number of registers.
- Uses overlapped register windows to speed - up procedure call and return.
- Efficient instruction pipeline.
- Efficient translation of high - level language programs into machine language programs by the compiler.

Example 6: An instruction set of a processor has 200 signals which can be divided into 5 groups of mutually exclusive signals as follows.

Group 1: 30 Signals
Group 2: 90 Signals
Group 3: 20 Signals
Group 4: 10 Signals
Group 5: 50 Signals

How many bits of the control words can be saved by using vertical micro-programming over horizontal microprogramming?

(A) 27 (B) 173
(C) 200 (D) 227

Solution: Horizontal micro-programming requires 200 signals. But vertical micro-programming uses encoding. So

Group 1 requires 5-bits ($\because 2^5 = 32$)
Group 2 requires 7-bits ($\because 2^7 = 128$)
Group 3 requires 5-bits ($\because 2^5 = 32$)
Group 4 requires 4-bits ($\because 2^4 = 16$)
Group 5 requires 6-bits ($\because 2^6 = 64$)

$\therefore$ Total bits required using vertical micro programming = 27

$\therefore$ Number of bits saved = 200 – 27 = 173

EXERCISE

Practice Problems I

Directions for questions 1 to 20: Select the correct alternative from the given choices.

1. Using two's complement arithmetic the resultant of 111100001111 – 110011110011 is
(A) 0010 0001 1111
(B) 0011 0000 1100
(C) 0010 0001 1101
(D) 0010 0001 1100

2. IEEE 32-bit floating point format of 384 is
(A) 0 10000111 00000000000000000000000
(B) 0 10000111 10000000000000000000000
(C) 0 00001000 00000000000000000000000
(D) 0 00001000 10000000000000000000000

3. Consider the following IEEE 32-bit floating point number:
0 01111110 10100000000000000000000.
What is the decimal value equivalent to given number?
(A) 0.25 (B) 3.25
(C) 0.8125 (D) 0.9375

4. What would be the bias value for a base-8 exponent in a 7-bit field?
(A) 8 (B) 16
(C) 63 (D) 64

5. The normalized value of the resultant of $8.844 \times 10^{-3} - 2.233 \times 10^{-1}$ is
(A) -2.144×10^{-1} (B) -0.2144
(C) -2×10^{-1} (D) -0.2

6. Which of the following is the correct sequence of micro-operations to add a number to the AC when the operand is a direct address operand and store the final result to AC?
(A) MAR←(IR(address))
MBR ← memory
$R_1 \leftarrow$ (AC) + (MBR)
(B) MAR ← IR(address)
MBR ← MAR
$R_1 \leftarrow$ (MBR)
$R_2 \leftarrow$ (AC) + (R_1)
AC ← R_2
(C) MAR ← (IR(address))
MBR ← Memory(MAR)
$R_1 \leftarrow$ (MBR)
$R_2 \leftarrow$ (AC) + (R_1)
AC ← (R_2)
(D) MAR ← (IR (address))
MBR ← Memory(MAR)
AC ← (AC) + (MAR)

Statement for linked answer questions 7 to 9: Assume that the control memory is 24 bits wide. The control portion of the micro-instruction format is divided into two fields. A micro-operation field of 13-bits specifies the micro-operation to be performed. An address selection field specifies a condition, based on the flags, that will cause a micro-instruction branch. There are eight flags.

7. How many bits are there in address selection field?
(A) 1 (B) 2
(C) 3 (D) 4

8. How many bits are there in address field?
(A) 8 (B) 9
(C) 13 (D) 24

9. What is the size of control memory in bits?
(A) 256 (B) 768
(C) 3328 (D) 6144

10. A simple processor has 3 major phases to its instructions cycle:
1. Fetch
2. Decode
3. Execute

Two 1-bit flags are used to specify the current phase in hardwired implementation. Will these flags required in micro-programming also?
(A) Yes
(B) No
(C) Cannot predict
(D) Depends on clock cycle time

11. In a 3-bus data path, the micro instructions format will be Opcode src1, src2, desti; The number of operations supported are 8 and the src1, src2 and desti require 20, 16 and 20 bits respectively.
The total number of horizontal microinstructions specified will be
(A) 2^{64} (B) 2^{8}
(C) 2^{56} (D) 2^{61}

12. What is the smallest positive normalized number represented using IEEE single precision floating point representation?
(A) 2^{-128} (B) $1 - 2^{-127}$
(C) 2^{-127} (D) 2^{-126}

13. A micro program control unit is required to generate a total of 30 control signals. Assume that during any micro instruction, almost two control signals are active. Minimum number of bits required in the control word to generate the required control signals will be
(A) 2 (B) 2.5
(C) 10 (D) 12

14. What is the fraction field of the single-precision floating point representation of 6.25?
(A) 1110 1000 0000 0000 0000 000
(B) 1001 0000 0000 0000 0000 000
(C) 1100 0000 0000 0000 0000 000
(D) 0110 0100 0000 0000 0000 000

15. Let the total number of control signals generated are n, then what is the number of bits allocated in control field of vertical micro programming?
(A) $n/2$ (B) n
(C) 2^n (D) $\log_2^n$

16. In a micro programmed control unit, a control field of one address control instruction has to support two groups of control signals. In group1 it is required to generate either one or none of the 32 control signals. In group 2 at most 5 from the remaining, what will be the number of bits needed for the control field?
(A) 8 (B) 10
(C) 35 (D) 37

17. Assume that the exponent e is constrained to lie in the range $0 \le e \le x$, with a bias of q, that the base is b and that the significant is P-digits in length.
What is the largest positive value that can be written is normalized floating point?
(A) $b^{x-q}(1 - b^{-p})$ (B) b^{-q-1}
(C) b^{-q-p} (D) $b^{x-q}(b^{-p} - 1)$

18. By using Booth's Multiplication algorithm. Below two numbers are multiplied:
Multiplicand: 0111 0111 1011 1101
Multiplier: 0101 1010 1110 1110
How many additions/subtractions are required for the multiplication of the above two numbers?
(A) 8 (B) 10
(C) 13 (D) 7

19. Let us assume, we are multiplying two positive integers 1101 and 1011. The multiplicand M is 1101 and Multiplier Q is 1011. What is partial product after second cycle?
(A) 0110 1101 (B) 1001 1110
(C) 0100 1111 (D) 1000 1111

20. The decimal representation of the 2's complement number 1101011 is
(A) 21 (B) −21
(C) 219 (D) 91

Practice Problems 2

Directions for questions 1 to 20: Select the correct alternative from the given choices.

1. A microprogrammed control unit
(A) is faster than a hard-wired control unit
(B) facilitates easy implementation of new instructions.
(C) is useful when very small programs are to be run.
(D) usually refers to the control unit of a micro-processor.

2. Micro-program is
(A) the name of a source program in micro computers.
(B) a primitive form of macros used in assembly language programming.
(C) a program of a very small size.
(D) the set of instruction indicating the basic elemental commands which directly control the operation of a system.

3. Programming that actually controls the path of signal or data within the computer is called
(A) System programming
(B) Micro-programming
(C) High-level language programming
(D) Assembly language programming

4. The instruction cycle time in a generic microprocessor is
(A) Longer than the machine cycle time
(B) Shorter than the machine cycle time
(C) Same as the machine cycle time
(D) Double the machine cycle time

5. Microprocessor unit or central processor unit consist of
(A) Control circuitry (B) ALU
(C) Memory (D) All of these

6. The exponent of a floating point number is represented in excess-N code so that
(A) the dynamic range is large
(B) overflow is avoided
(C) the precision is high
(D) the smallest number is represented efficiently

7. Using Booth's algorithm for Multiplication, the Multiplier −14 is coded as
(A) 11110 (B) 01110
(C) 10010 (D) 00010

8. Data Path consists of
(A) Registers (B) ALU
(C) Bus (D) All of these

9. A floating point number that has a '0' in MSB of mantissa is said to have ____
(A) Overflow (B) Underflow
(C) Normalization (D) Positive exponent

10. Let the Binary sum after BCD addition is stored in K, Z_8, Z_4, Z_2, and Z_1 Then the condition for a correction and output carry can be expressed as $C =$
(A) $K + Z_8 Z_4 + Z_8 Z_2$ (B) $K + Z_8 Z_4 + Z_4 Z_2$
(C) $K + Z_8 Z_2 + Z_8 Z_1$ (D) $K + Z_4 Z_2 + Z_2 Z_1$

11. Which of the following is an advantage of biased exponents?
(A) Convenient way to represent exponents
(B) Useful for conversion
(C) Convenient for comparison purposes
(D) All of these

12. Booth multiplication skips over runs of zeros and ones which reduces the number of add and subtract steps needed to multiply two n-bit numbers to n to a variable number whose average value n_{avg} is less than n what will be n_{avg}?
(A) $n/3$ (B) $n/4$
(C) $n/2$ (D) n

13. The sequence of events that happen during a fetch operation is:
(A) PC → memory → IR
(B) PC → MAR → memory → IR
(C) PC → MAR → memory → MDR → IR
(D) PC → memory → MDR → IR

14. Micro-programming is a technique for
(A) Programming input or output routines
(B) Programming the microprocessors
(C) Programming the control steps of a computer
(D) Writing small programs

15. In a micro program ____ specifies the address of Micro-instructions to be executed.
(A) AR (B) PC
(C) SP (D) CAR

16. Which one of the following statements is correct?
(A) Micro-programmed control unit is costlier and slow.
(B) Micro-programmed control unit are cheap and slow.
(C) Micro-programmed control unit is costlier and fast.
(D) Micro-programmed control unit are fast and cheaper.

17. Horizontal micro-instructions have
(A) High degree parallelism, more encoding of control information.
(B) High degree parallelism, little encoding of control information.
(C) Low degree parallelism, more encoding of control information.
(D) Low degree parallelism, little encoding of control information.

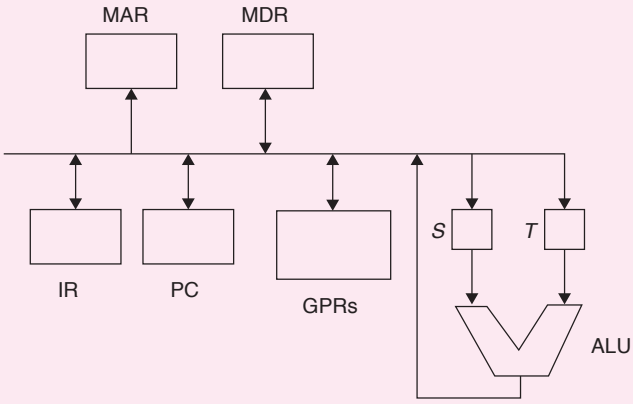
18. A vertical micro-instruction have ____.
(A) Short formats and considerable encoding of control information
(B) Long formats and considerable encoding of control information
(C) Short formats and little encoding of control information
(D) Long formats and little encoding of control information

19. Guard bits are used to
(A) avoid unnecessary loss of MSB
(B) avoid unnecessary loss of LSB
(C) the loss of MSB
(D) the loss of LSB

20. Which of the following is not the essential element of a number represented in floating-point notation?
(A) Exponent (B) Significand
(C) Sign (D) Normalization

Previous Years' Questions

Common data for questions 1 and 2: Consider the following data path of a CPU.



The ALU, the bus and all the registers in the data path are of identical size. All operations including incrementation of the PC and the GPRs are to be carried out in the ALU. Two clock cycles are needed for memory read operation—the first one for loading address in the MAR and the next one for loading data from the memory bus into the MDR.

1. The instruction 'add R_0, R_1' has the register transfer interpretation $R_0 \leftarrow R_0 + R_1$. The minimum number of clock cycles needed for execution cycle of this instruction is

(A) 2 (B) 3
(C) 4 (D) 5

2. The instruction 'call *Rn*, sub' is a two word instruction. Assuming that PC is incremented during the fetch cycle of the first word of the instruction, its register transfer interpretation is

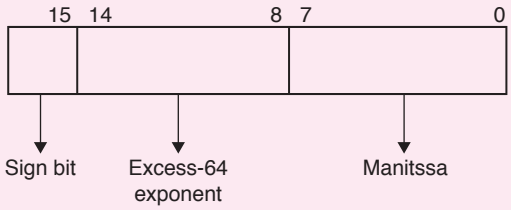
Rn ← PC + 1;

PC ← M[PC]

The minimum number of CPU clock cycles, needed during the execution cycle of this instruction is

(A) 2 (B) 3
(C) 4 (D) 5

Data for question 3: Consider the following floating-point format.



Mantissa is a pure fraction in sign-magnitude form.

3. The normalized representation for the above format is specified as follows. The mantissa has an implicit 1 preceding the binary (radix) point. Assume that only 0 's are padded in while shifting a field.

The normalized representation of the above number (0.239×2^{13}) is: **[2005]**

(A) 0*A* 20 (B) 11 34
(C) 49 *D*0 (D) 4*A E*8

4. In the IEEE floating point representation the hexadecimal value 0*x*00000000 corresponds to **[2008]**

(A) The normalized value 2^{-127}
(B) The normalized value 2^{-126}
(C) The normalized value + 0
(D) The special value + 0

5. *P* is a 16-bit signed integer. The 2's complement representation of *P* is $(F87B)_{16}$. The 2's complement representation of 8**P* is **[2010]**

(A) $(C3D8)_{16}$ (B) $(187B)_{16}$
(C) $(F878)_{16}$ (D) $(987B)_{16}$

6. The decimal value 0.5 in IEEE single precision floating point representation has **[2012]**

(A) fraction bits of 000 … 000 and exponent value of 0
(B) fraction bits of 000…000 and exponent value of –1
(C) fraction bits of 100…000 and exponent value of 0
(D) no exact representation

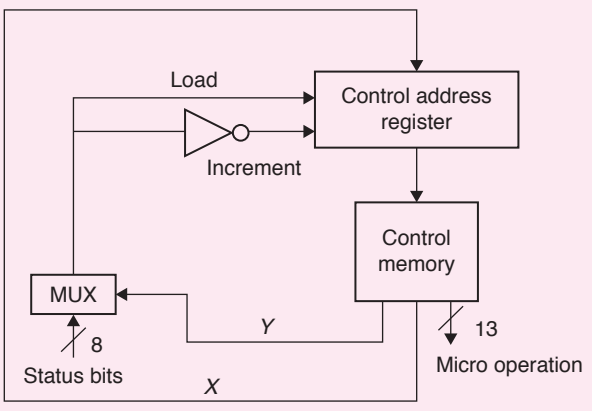
7. The smallest integer that can be represented by an 8-bit number in 2's complement form is **[2013]**

(A) –256 (B) –128
(C) –127 (D) 0

8. Let *A* = 1111 1010 and *B* = 0000 1010 be two 8-bit 2's complement numbers. Their product in 2's complement is **[2004]**

(A) 1100 0100 (B) 1001 1100
(C) 1010 0101 (D) 1101 0101

9. The microinstructions stored in the control memory of a processor have a width of 26 bits. Each microinstruction is divided into three fields, a micro-operation field of 13 bits, a next address field (*X*), and a MUX select field (*Y*), there are 8 status bits in the inputs of the MUX **[2004]**



How many bits are there in the X and Y fields, and what is the size of the control memory in number of words?

(A) 10, 3, 1024 (B) 8, 5, 256
(C) 5, 8, 2048 (D) 10, 3, 512

10. Consider the following sequence of micro-operations.

MBR ← PC
MAR ← X
PC ← Y
Memory ← MBR

Which one of the following is a possible operation performed by this sequence? **[2013]**

(A) Instruction fetch
(B) Operand fetch
(C) Conditional branch
(D) Initiation of interrupt service

11. For computers based on three-address instruction formats, each address field can be used to specify which of the following: **[2015]**

(S_1) A memory operand
(S_2) A processor register
(S_3) An implied accumulator register

(A) Either S_1 or S_2
(B) Either S_2 or S_3
(C) Only S_2 and S_3
(D) All of S_1, S_2 and S_3

12. Let X be the number of distinct 16 - bit integers in 2's complement representation. Let Y be the number of distinct 16 - bit integers in sign magnitude representation. They $x - y$ is _____ . **[2016]**

13. The n-bit fixed-point representation of an unsigned real number X uses f bits for the fraction part. Let $i = n - f$. The range of decimal values for X in this representation is **[2017]**

(A) 2^{-f} to 2^i (B) 2^{-f} to $(2^i - 2^{-f})$
(C) 0 to 2^i (D) 0 to $(2^i - 2^{-f})$

14 Consider the C code fragment given below.

```
typedef struct node  {
  int data;
  node* next;
}   node;
void join (node* m, node* n)  {
  node* p = n;
  while (p - >next != NULL)  {
    p = p - >next;
  }
  p - >next = m;
}
```

Assuming that m and n point to valid NULL-terminated linked lists, invocation of join will **[2017]**

(A) append list m to the end of list n for all inputs.
(B) either cause a null pointer dereference or append list m to the end of list n.
(C) cause a null pointer dereference for all inputs.
(D) append list n to the end of list m for all inputs.

15. The representation of the value of a 16-bit unsigned integer X in hexadecimal number system is BCA9. The representation of the value of X in octal number system is **[2017]**

(A) 571244 (B) 736251
(C) 571247 (D) 136251

16. Consider the following processor design characteristics.

I. Register-to-register arithmetic operations only
II. Fixed-length instruction format
III. Hardwired control unit

Which of the characteristics above are used in the design of a RISC processor? **[2018]**

(A) I and II only (B) II and III only
(C) I and III only (D) I, II and III

17. Consider the unsigned 8-bit fixed point binary number representation below:

$$b_7\, b_6\, b_5\, b_4\, b_3 \cdot b_2\, b_1\, b_0$$

where the position of the binary point is between b_3 and b_2. Assume b_7 is the most significant bit. Some of the decimal numbers listed below cannot be represented exactly in the above representation:

(i) 31.500 (ii) 0.875
(iii) 12.100 (iv) 3.001

Which one of the following statements is true? **[2018]**

(A) None of (i), (ii), (iii), (iv) can be exactly represented
(B) Only (ii) cannot be exactly represented
(C) Only (iii) and (iv) cannot be exactly represented
(D) Only (i) and (ii) cannot be exactly represented

Answer Keys

Exercises

Practice Problems I

1. D	**2.** B	**3.** C	**4.** C	**5.** A	**6.** C	**7.** C	**8.** A	**9.** D	**10.** B
11. A	**12.** D	**13.** C	**14.** B	**15.** D	**16.** B	**17.** A	**18.** B	**19.** B	**20.** B

Practice Problems I

1. B	**2.** D	**3.** B	**4.** C	**5.** D	**6.** D	**7.** C	**8.** D	**9.** B	**10.** A
11. C	**12.** C	**13.** C	**14.** C	**15.** D	**16.** A	**17.** B	**18.** A	**19.** B	**20.** D

Previous Years' Questions

1. B	**2.** B	**3.** D	**4.** D	**5.** A	**6.** B	**7.** B	**8.** A	**9.** A	**10.** D
11. A	**12.** 1	**13.** D	**14.** B	**15.** D	**16.** D	**17.** C			

Chapter 3

Memory Interface, I/O Interface

LEARNING OBJECTIVES

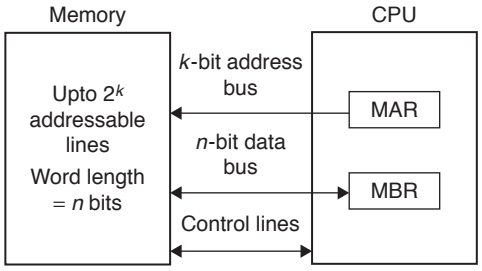
- *Memory interface*
- *RAM*
- *ROM*
- *Memory interfacing*
- *Input–output interfacing*
- *Handshaking*
- *Data transfer mode*
- *Design techniques for interrupts*
- *Direct memory access*
- *Input–output processor*

Memory Interface

Basic Concepts

Computer memory is used to store programs and data. The maximum size of a memory that can be used in any computer is determined by the addressing scheme.

Example: If the memory address has 16-bits, then the size of memory will be 2^{16} Bytes.



If MAR is k-bits long and MDR is n-bits long, then the memory may contain up to 2^k addressable locations and the n-bits of data are transferred between the processor and memory. This transfer takes place over processor bus. The processor bus has

1. Address line
2. Data line
3. Control line

Control line is used for coordinating data transfer.

Processor reads the data from the memory by loading the address of the required memory location into MAR and setting the $R/\overline{W}$ line to 1.

The memory responds by placing the data from the addressed location onto the data lines and confirms the actions. Upon confirmation, the processor loads the data onto the data lines, into MDR register. The processor writes the data into the memory location by loading the address of this location into MAR and loading the data into MDR sets the $R/\overline{W}$ line to 0.

- **Memory Access Time:** It is the time that elapses between the initiation of an operation and the completion of that operation.
- **Memory Cycle Time:** It is the minimum time delay that required between the initiations of two successive memory operations.

RAM (Random Access Memory)

In RAM, if any location that can be accessed for a read/write operation in fixed amount of time, it is independent of the location's address:

- Memory cells are usually organized in the form of array, in which each cell is capable of storing one bit of information.
- Each row of cells constitutes a memory word and all cells of a row are connected to a common line called as word line.
- The cells in each column are connected to sense/write circuit by two bit lines.

The data input and data output of each sense/write circuit are connected to a single bidirectional data line that can be connected to a data bus.

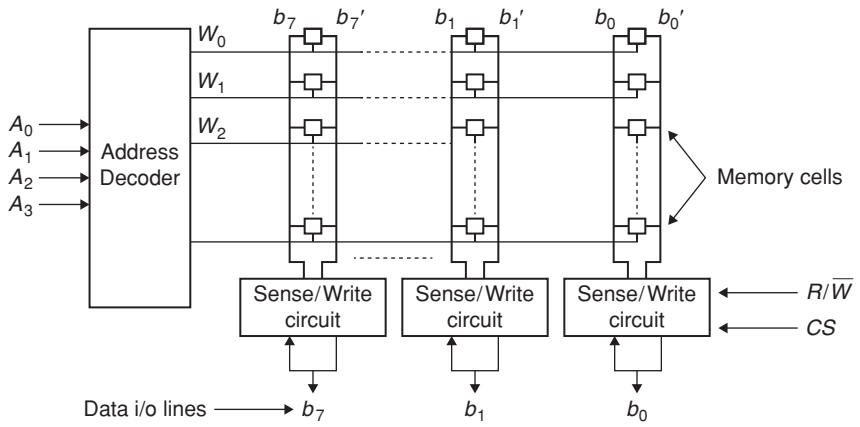


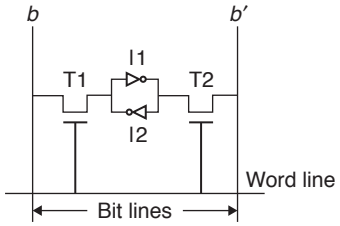
Figure 1 Organization of bit cells in a memory chip

- $R/\overline{W}$: Specifies the required operation.
- CS: Chip select input selects a given chip in the multi-chip memory system.

Static memories

Memories that consist of circuits capable of retaining their state as long as power is applied are known as static memories.

SRAM (static RAM) SRAM consists of two inverters, two transistors. In order to read the state of the SRAM cell, the word line is activated to close switches T1 and T2.



Advantages of SRAM:

1. It has low power consumption, because the current flows in the cell only when the cell is being activated or accessed.
2. SRAM can be accessed quickly.

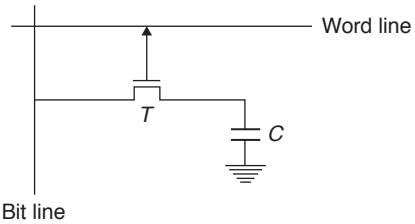
Disadvantages of SRAM: SRAMs are said to be volatile memories, because their contents are lost when the power is interrupted.

DRAM (Dynamic RAM) Less expensive RAMs can be implemented if simplex cells are used, such cells cannot retain their state indefinitely. Hence they are called dynamic RAMs.

The information stored in a dynamic memory cell in the form of a charge on a capacitor and this charge can be maintained only for tens of milliseconds.

The contents must be periodically refreshed by restoring the capacitor charge to its full value.

Example: Single-transistor dynamic memory cell:



If charge on capacitor > threshold value, then bit line will have '1'. If charge on capacitor < threshold value, then bit line will have '0'.

DRAM	SRAM
1. Volatile	1. Volatile
2. Simple to build and slower than SRAM	2. Faster than DRAM
3. Need refresh circuitry	3. More expensive to build
4. Favoured for large memory units	4. Favoured for cache memory

Latency It is the amount of time it takes to transfer a word of data to or from the memory.

- For the transfer of a single word, the latency provides the complete indication of memory performance.
- For a block transfer, the latency denotes the time it takes to transfer the first word of data.

Bandwidth It is defined as the number of bits or bytes that can be transferred in one second.

Note: All dynamic memories have to be refreshed.

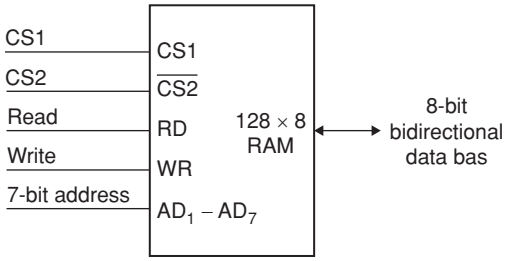


Figure 2 RAM chip block diagram

Read-only Memory (ROM)

Both SRAM and DRAM chips are volatile, which means that they lose the stored information if power is turned off. If the normal operation involves only reading of stored data, use ROM memory.

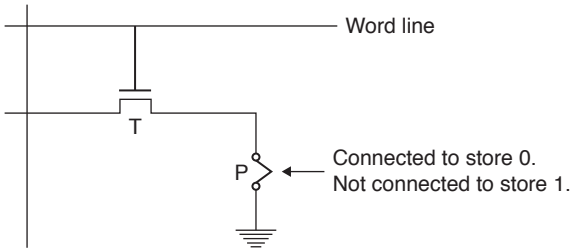


Figure 3 ROM cell

Types of ROM

Different types of non-volatile ROM are:

1. **PROM (Programmable ROM):**
   - Allows the data to be loaded by the user.
   - Less expensive, faster, flexible.
2. **EPROM (Erasable PROM):**
   - Allows the stored data to be erased and new data to be loaded.
   - Flexible, retain information for a long time.
   - Contents erased by UV light.
3. **EEPROM (Electrically Erasable PROM):**
   - Programmed and erased electrically.
   - Allows the erasing of all cell contents selectively.
   - Requires different voltage for erasing, writing and reading of stored data.
4. **Flash memory:** Allows to read the contents of a single cell but it is only possible to write the entire contents of a block.

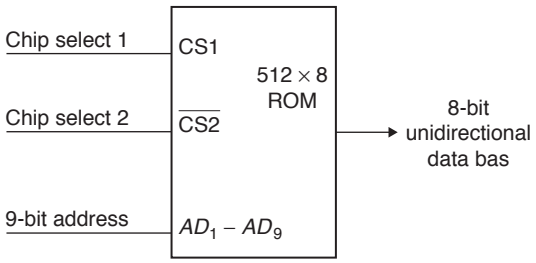


Figure 4 Block diagram of ROM chip

Memory Interfacing

The interfacing circuit enables the access of processor to memory. The function of memory interfacing is that the processor should be able to read from and write into a given register of a memory chip. To perform this, the microprocessor should be

1. able to select the chip.
2. identify the register.
3. enable the appropriate buffer.

Input–output Interfacing

Basic Concepts of I/O Module

I/O module contains logic for performing a communication function between the peripherals and the bus. The peripherals are not connected to the system bus directly. The reasons for this are

1. Peripherals are electromechanical and electromagnetic devices and their manner of operation is different from the operation of the CPU and memory, which are electronicdevices. So a conversion of signal values may be required.
2. The data transfer rate of peripherals is usually slower than the transfer rate of the CPU and hence a synchronization mechanism may be needed.
3. Data codes and formats in peripherals differ from the word format in the CPU and memory.
4. The operating modes of peripherals are different from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

To resolve these differences, computer systems include special hardware components between the CPU and peripherals to supervise and synchronize all input and output transfers. These components are called 'interface' units.

By using this interfacing,

1. interface to the processor and memory via the system bus or central switch.
2. interface to one or more peripheral devices by tailored data links.

Input–output devices

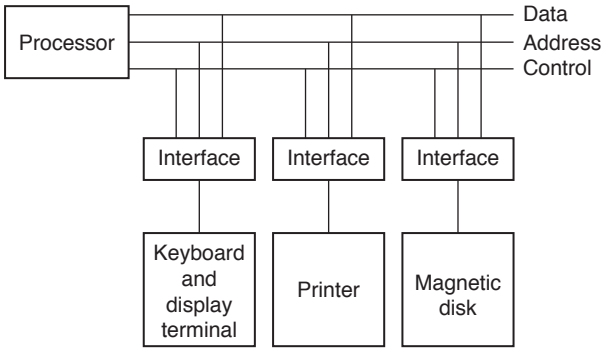
- Input and Output devices provide a means for people to make use of a computer.
- Some I/O devices function as an interface between a computer system and other physical system.

Input–output interface

Input/output Interface provides a method for transferring information between internal storage (such as memory and CPU Register) and external I/O devices. It resolves the difference between the computer and peripheral devices.

Input–output bus and interface modules

Each peripheral has an interface module associated with it. The interface module decodes the device address (device code), decodes signals for the peripheral controller, synchronizes the data flow and supervises the transfer rate between peripheral and CPU or memory.



Function of buses

1. **Memory bus:** It is used for information transfer between CPU and main memory.
2. **I/O bus:** It is used for information transfers between CPU and I/O devices through their I/O interface.

Isolated versus memory mapped I/O

1. **Isolated I/O:**
   - Separate I/O read/write control lines in addition to memory read/write control lines.
   - Separate (isolated) memory and I/O address space
   - Distinct input and output instructions.
2. **Memory-mapped I/O:**
   - A single set of Read/write control lines (i.e., no distinction between memory and I/O transfer).
   - Memory and I/O address share the common address space (reduces memory address range available).
   - No specific input or output instruction.
   - The same memory reference instructions can be used for I/O transfer.
   - Considerable flexibility in handling I/O operations.

Asynchronous serial transfer

In serial data transmission, each bit in the message is sent in sequence one at a time. Serial transmission can be synchronous or asynchronous.

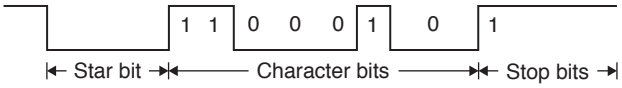
In synchronous transmission, the two units share a common clock frequency and bits are transmitted continuously at the rate dictated by the clock pulses.

In asynchronous transmission, binary information is sent only when it is available and the line remains idle when there is no information to be transmitted.

In serial asynchronous transmission technique, each character consists of three parts:

1. start bits
2. character bits
3. stop bits

Example:



A transmitted character can be detected by the receiver from the knowledge of the transmission rules:

1. When a character is not being sent, the line is kept in the 1-state.
2. The initiation of a character transmission is detected from the start bit, which is always 0.
3. The character bits always follow the start bit.
4. After the last bit of the character is transmitted, a stop bit is detected when the line returns to the 1-state for at least one bit time.
   - The baud rate is defined as the rate at which serial information is transmitted and is equivalent to the data transfer in bits per second.

Strobe control Employs a single control line to time each transfer. Strobe may be activated by either the source or the destination units.

(a) Source initiated transfer:

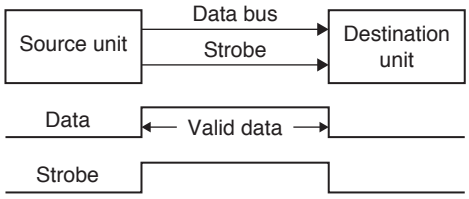
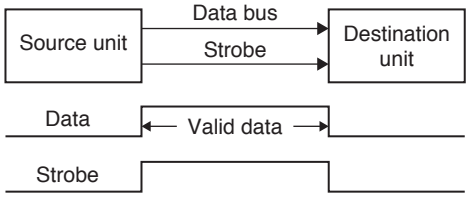


Figure 5 Source initiated strobe for data transfer

- The data bus carries the binary information from source unit to the destination unit.
- The strobe is a single line that informs the destination unit when a valid data word is available in the bus.

(b) Destination initiated strobe for data transfer:



- The destination unit activates the strobe pulse, informing the source to provide the data. The source unit responds by placing the requested binary information on the data bus.
- The data must be valid and remain in the bus long enough for the destination unit to accept it.
- The falling edge of the strobe pulse can be used again to trigger a destination register. The destination unit then disables the strobe.

Handshaking

Disadvantage of strobe method: Source unit which initiated the transfer has no way of knowing whether the destination unit has actually received the data item that was placed in the bus.

The handshake method solves this problem by introducing a second control signal that provides a reply to the unit that initiates the transfer.

Principle of two-wire handshaking: One control line is in the same direction as the data flow in the bus from the source to the destination. It is used by the source unit to inform the destination unit whether there are valid data in the bus.

The other control line is in the other direction from the destination to the source. It is used by the destination unit to inform the source whether it can accept data.

The sequence of control during the transfer depends on the unit that initiates the transfer.

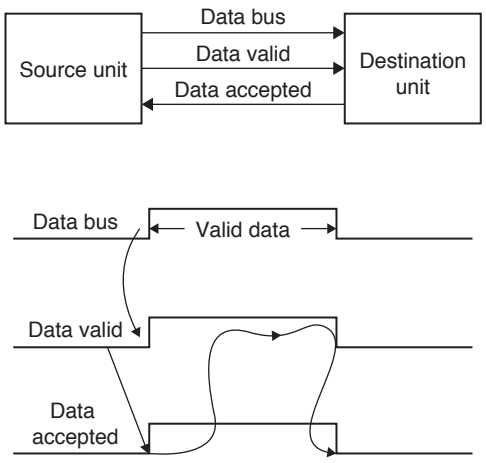


Figure 6 Source initiated transfer using hand shaking

Similarly a destination unit may also initiate the transfer.

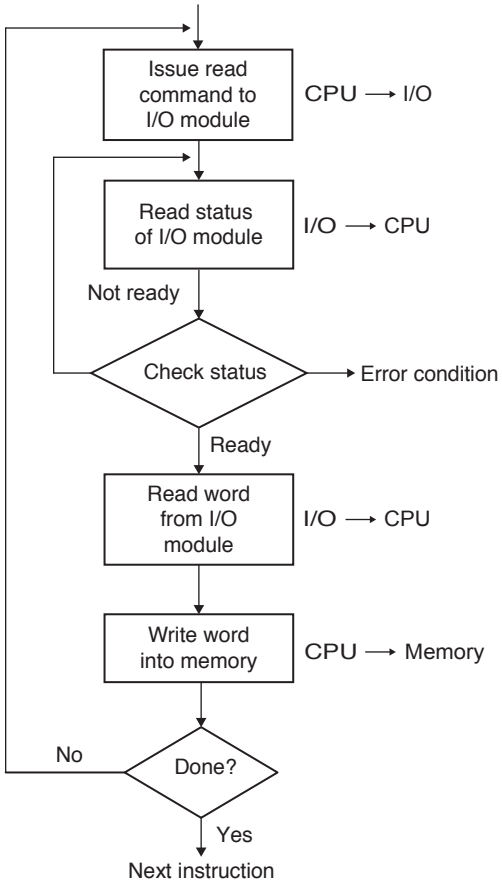
Advantage: Handshaking scheme provides a high degree of flexibility and reliability because the successful completion of a data transfer relies on active participation by both units.

Modes of transfer

There are three different data transfer modes between the central computer (CPU or Memory) and peripherals:

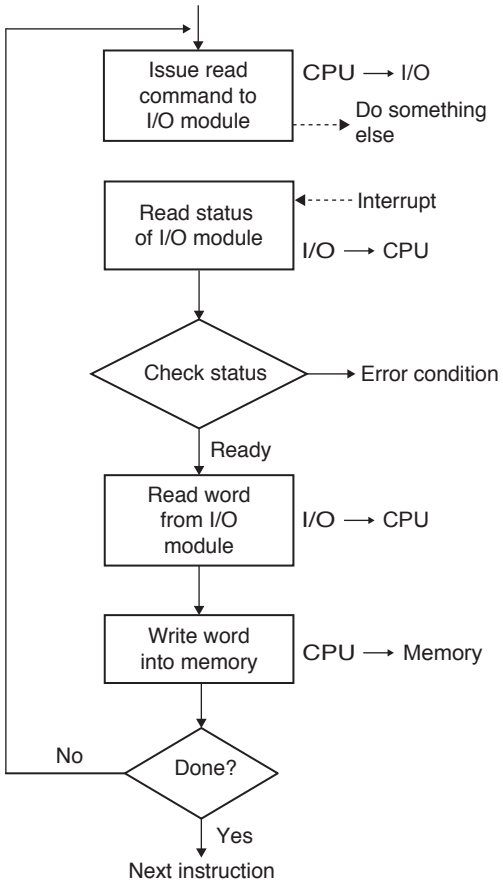
1. Program-controlled I/O
2. Interrupt-initiated I/O
3. Direct memory access

Program-controlled input–output With programmed I/O, the I/O module will perform the requested action and then set the appropriate bits in the I/O status register. The I/O module takes no further action to alert the CPU. In particular it does not interrupt the CPU. Thus, it is the responsibility of the CPU to periodically check the status of the I/O module until it finds that the operation is complete.



Interrupt initiated input–output The problem with programmed I/O is that the CPU has to wait a long time for the I/O module of concern to be ready for either reception or transmission of data. The CPU, while waiting must repeatedly interrogate the status of the I/O module. As a result, the level of the performance of the entire system is severely degraded.

An alternation is for the CPU to issue an I/O command to a module and then go on to do some other useful work. The I/O module will then interrupt the CPU to request service when it is ready to exchange data with the CPU. The CPU then executes the data transfer, as before and then resumes its former processing.



Interrupt Processing: In all computers, there is a mechanism by which the normal processing of the processor is interrupted by other modules like I/O, memory. The interrupts may be of the following class:

1. Program: Generated by some condition that occurs as a result of an instruction execution.

   Examples: Arithmetic overflow, division by zero, etc.
2. Timer: Generated by timer within the processor.
3. I/O: Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.
4. Hardware failure: Generated by a failure such as power failure or memory parity error.

Interrupts are provided primarily as a way to improve processing efficiency.

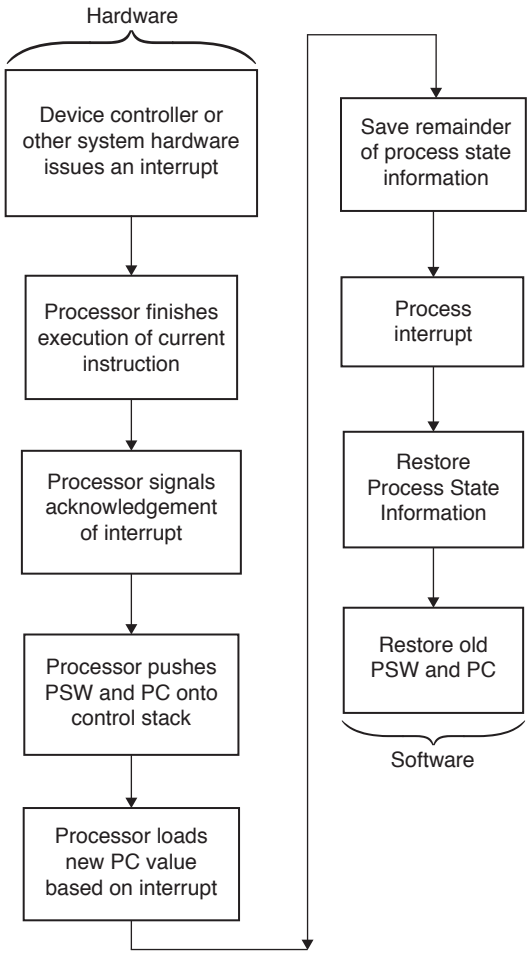


Figure 7 Interrupt Processing Flowchart

Consider the following figures, which show the contents of memory and registers before and after interrupt instruction processing.

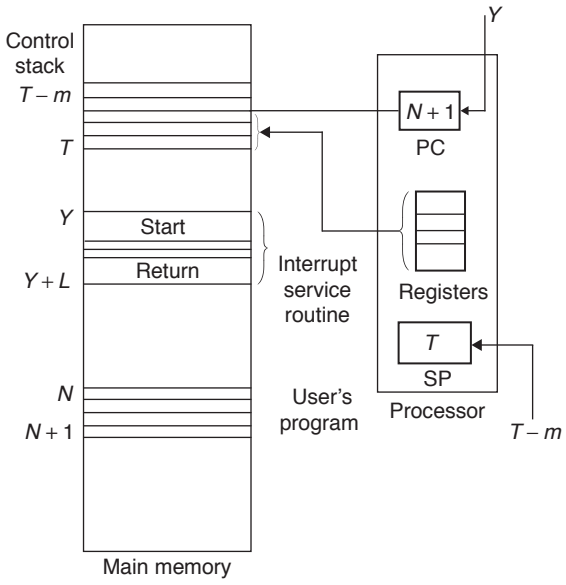


Figure 8 Interrupt occurs after instruction at location *N*

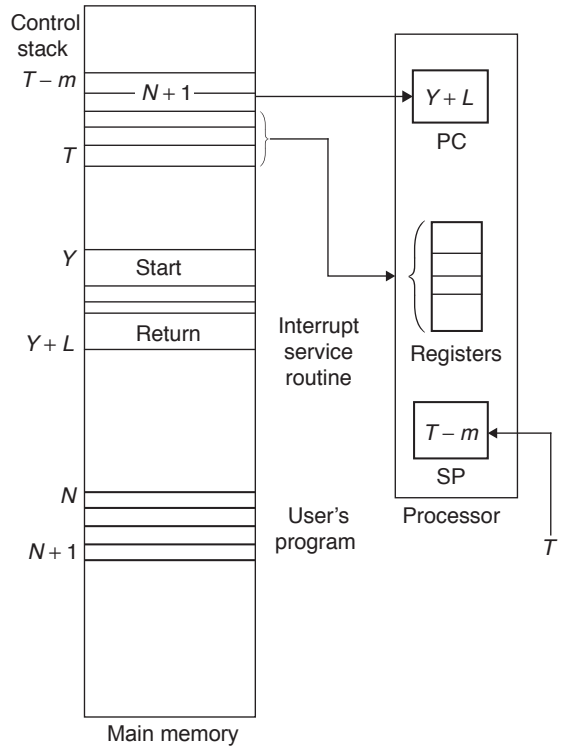


Figure 9 Return from interrupt

Interrupt priority: Priority determines which interrupt is to be served first when two or more requests are made simultaneously. Priority also determines which interrupts are permitted to interrupt the computer while another is being serviced. Higher priority interrupts can make requests while servicing a lower priority interrupt.

Design techniques for interrupts: Two design issues arise in implementing interrupt I/O:

1. Since there will almost invariably be multiple I/O module, how does the CPU determine which device issued the interrupt.
2. If multiple interrupts have occurred, how does the CPU decide which one to process.

Four general categories of techniques are there which are common in use:

(a) Multiple interrupt lines: In this technique, multiple interrupt lines are provided between the CPU and the I/O modules. However, it is impractical to dedicate more than a few bus lines or CPU pins to interrupt lines. Consequently, even if multiple lines are used, it is likely that each line will have multiple I/O modules attached to it. Thus, one of the other three techniques must be used one each line.

(b) Software poll: When the CPU detects an interrupt, it braches to an interrupt-service routine whose job is to poll each I/O module to determine which module generated the interrupt. The poll could be in the form of a separate command line. The CPU receives the command and places the address of a particular I/O module on the address lines. The I/O module responds positively if it set the interrupt. Alternatively, each I/O module could contain an addressable status register. The CPU then read the status register of each I/O module to identify the interrupting module. Once the correct module is identified, the CPU branches to a device service routine specified to that device. It is time consuming.

(c) Daisy chain: Daisy chain in effect provides a hardware poll. For interrupts all I/O modules share a common interrupt request line. The interrupt acknowledge line is daisy chained through the modules. When the CPU is interrupted, it sends out an interrupt acknowledgement. This signal propagates through a series of I/O modules until it gets to a requesting module. The requesting module typically responds by placing a word on the data lines. This word is referred to as a vector and is either the address of the I/O module or some other unique identifier. In either case, the CPU uses the vector as a pointer to the appropriate device-service routine. This avoids the need to execute a general interrupt-service routine first. This technique is referred to as a vectored Interrupt.

(d) Bus arbitration: Bus arbitration is also another technique which makes use of vectored Interrupts. With bus arbitration, an I/O module must first gain control of the bus before it can raise the interrupt request line. Thus only one module can raise the line at a time. When the CPU detects the interrupt, it responds on the interrupt acknowledge line. The requesting module then places its vector on the data lines.

Direct Memory Access (DMA)

Drawbacks of programmed and interrupt-driven I/O:

1. The I/O transfer rate is limited by the speed with which the processor can test and service a device.
2. The processor is tied up in managing I/O transfer; a number of instructions must be executed for each I/O transfer.

Both methods have an adverse impact on both processor activity and I/O transfer rate.

When large volume of data is to be moved, a more efficient technique is required: Direct Memory Access (DMA).

DMA function: DMA involves an additional module on the system bus.

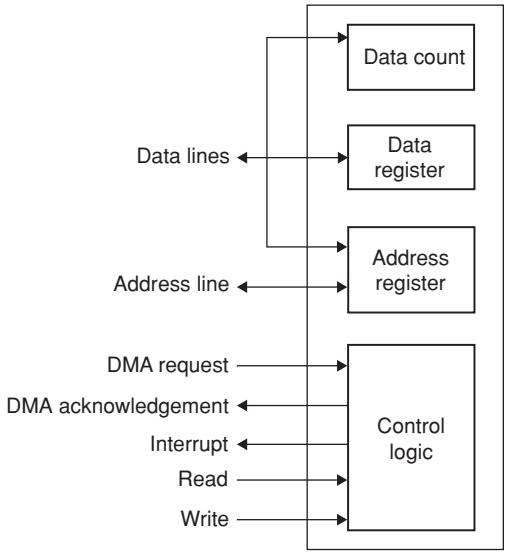
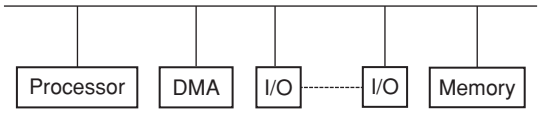


Figure 10 DMA Block Diagram

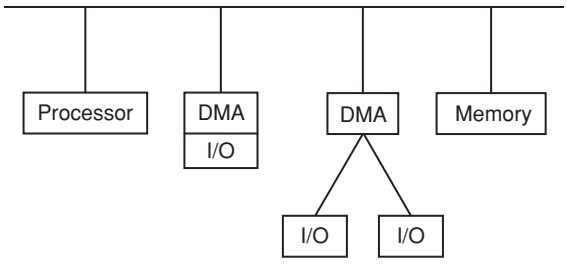
The DMA module is capable of mimicking the processor and indeed, of taking over control of the system from the processor. It needs to do this to transfer data to and from memory over the system bus. For this purpose, the DMA module must use the bus only when the processor does not need it or it must force the processor to suspend operation temporarily. The latter technique is more common and is referred as cycle stealing.

DMA configurations

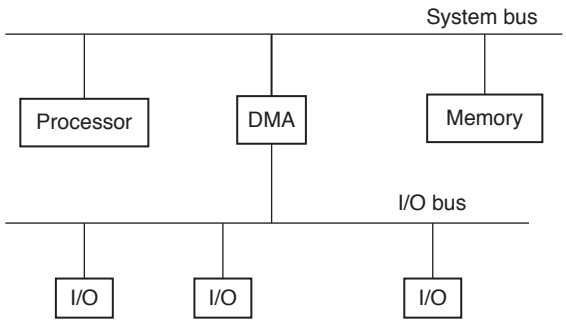
1. **Single bus, detached DMA**
   - Inexpensive, inefficient
   - Each transfer of a word consumes two bus cycles.



2. **Single bus, integrated DMA I/O** There is a path between the DMA module and one or more I/O modules that does not include system bus.



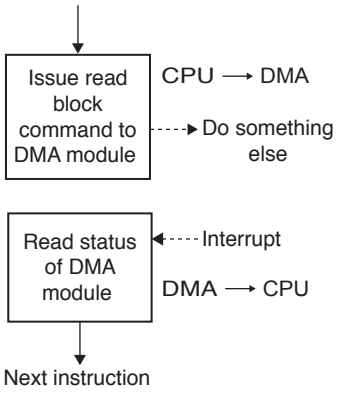
3. **I/O bus**
   - Reduces the number of I/O interfaces in the DMA module to one.
   - Easily expandable configuration.



With DMA, when the CPU wishes to read or write a block of data, it issues a command to the DMA module, by sending the following information to the DMA module.

1. Whether a read or write is requested.
2. The address of the I/O device involved.
3. The starting location in memory to read from or write to.
4. The number of words to be read or written.

The CPU then continues with other work. It has delegates this I/O operation to the DMA module, and that module will take care of it. The DMA module transfers the entire block of data, one word at a time, directly to or from memory, without going through the CPU. When the transfer is complete, the DMA module sends an interrupt signal to the CPU. Thus, the CPU is involved only at the beginning and end of the transfer.



DMA transfer can either happen as:

1. **Burst transfer**
   - A block sequence consisting of a number of memory words is transferred in continuous burst.
   - DMA controller is master of memory Buses.
   - This mode of transfer is needed for fast devices such as Magnetic Disks, where transmission cannot be stopped or slowed down.
2. **Cycle stealing**
   - CPU is usually much faster than I/O (DMA), thus CPU uses the most of the memory cycles.
   - DMA controller steals the memory cycles from CPU.
   - For those stolen cycles, CPU remains idle.

- For those slow CPU, DMA Controller may steal most of the memory.
- Cycle stealing, which may cause CPU remain idle long time.

Input–Output Processor (IOP)

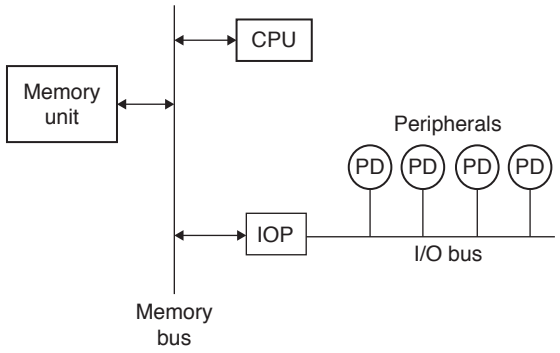
An IOP is a processor, having a direct memory access capability, used to communicate with I/O devices.

In this configuration, the computer system can be divided into a memory unit and a number of processors comprised of the CPU and one or more IOPs.

Each IOP takes care of input and output tasks, relieving the CPU from the house keeping chores involved in I/O transfers.

- IOP is similar to a CPU except that it is designed to handle the details of I/O processing.
- Unlike DMA, the IOP can fetch and execute its own instructions.

The following figure shows a computer with two processors:



Exercises

Practice Problems I

Directions for questions 1 to 20: Select the correct alternative from the given choices.

1. Consider a DRAM that must be given a refresh cycle 64 times per ms. Each refresh operation requires 150 ns, a memory cycle requires 250 ns. What is the approximate percentage of the memory's total operating time must be given to refreshes?
(A) 1% (B) 2%
(C) 9% (D) 60%

2. A DMA controller transfers 16-bit words to memory using cycle stealing. The words are assembled from a device that transmits characters at a rate of 2400 characters per second. The CPU is fetching and executing instructions at an average rate of 1 million instructions per second. By how much time will the CPU be slowed down because of the DMA transfer?
(A) 0.6% (B) 0.1%
(C) 0.12% (D) 0.24%

3. A system is based on a 16-bit microprocessor and has two I/O devices. The I/O controllers for this system use separate control and status registers. Both devices handle data on a one-byte-at-a time basis. The first device has two status lines and three control lines. The second device has three status lines and four control lines. How many 16-bit I/O control module registers do we need for status reading and control of each device?
(A) 1, 2 (B) 2, 1
(C) 2, 2 (D) 1, 1

4. In a programmed I/O technique, the processor is stuck in a wait loop doing status checking of an I/O device. To increase efficiency, the I/O software could be written so that the processor periodically checks the status of the device. If the device is not ready, the processor can jump to other tasks. After some timed interval, the processor comes back to check status again. Let us assume that above scheme is used for outputting data one character at a time to a printer that operates at 10 characters per second (CPS). Which of the following statement is true if its status is scanned every 200 ms?
(A) The printing speed is increased by 5 CPS
(B) The printing rate is slowed to 5 CPS
(C) The printing rate is at 10 CPS only
(D) The printing rate is at 20 CPS

5. Consider a system employing interrupt-driven I/O for a particular device that transfers data at an average of 8 KB/s on a continuous basis. The interrupt processing takes about 100 μs and the I/O device interrupts processor for every byte. Let assume that the device has two 16-byte buffers and interrupts the processor when one of the buffer is full. While executing the ISR, the processor takes about 8 μs for the transfer of each byte. Then what is the fraction of processor time is consumed by this I/O device?
(A) 8% (B) 11%
(C) 50% (D) 65%

6. A 32-bit computer has two selector channels one multiplexor channel. Each selector channel supports two magnetic disks and three magnetic tape units. The multiplexor channel has two line printers, two card readers and 10 VDT terminals connected to it. Assume the following transfer rates:
Disk drive: 1000 KB/sec
Magnetic tape drive: 300 KB/sec
Line printer: 6.2 KB/sec
Card reader: 2.4 KB/sec
VDT: 1 KB/sec

What is the maximum aggregate I/O transfer rate of this system?
(A) 1625.6 KB/s (B) 1327.2 KB/s
(C) 2027.2 KN/s (D) 2327.2 KB/s

7. Consider a disk drive with 16 surfaces, 512 tracks per surface and 512 sectors per track, 1 kilo bytes per sector and a Rotation speed of 3000 RPM. The disk is operated in cycle stealing mode where by whenever one 4 byte word is ready it is sent to memory; similarly, for writing, the disk interface read a 4 byte word from the memory in each DMA cycle. The memory cycle time is 40 nsec. Find the maximum percentage of time that the CPU gets blocked during DMA operation?
(A) 2.62% (B) 26.21%
(C) 0.26% (D) 0.52%

8. How many RAM chips of size (256 K × 1-bit) are needed to build a 1 M Byte memory?
(A) 16 (B) 8
(C) 32 (D) 24

9. Four memory chips of 16 × 4 size have their address bases connected together. The whole system will have a size of
(A) 16 × 8 (B) 64 × 64
(C) 16 × 16 (D) 256 × 1

10. In which of following I/O techniques, there will be no interrupt?
(A) Programmed I/O (B) Interrupt-driven I/O
(C) DMA (D) Both (B) and (C)

11. The capacity of a memory unit is defined by the number of words multiplied by the number of bits/word. How many separate address and data lines are needed for a memory 16K × 16?
(A) 10 address, 4 data lines
(B) 14, 4
(C) 14, 16
(D) 14, 14

12. The main problem of strobe asynchronous data transfer is
(A) it employs a single control line
(B) it is controlled by clock pulses in the CPU.
(C) the falling edge again used to trigger
(D) no way of knowing whether the destination has received the data item.

13. Which of the following DMA transfer modes and interrupt handling mechanisms will enable the highest I/O bandwidth?
(A) Block transfer and polling interrupt
(B) Cycle stealing and polling interrupt
(C) Block transfer and vectored interrupt
(D) Transparent DMA and vectored interrupt

14. Which of the following enables peripherals to pass a signal down the bus to the next device on the bus during polling of the device?
(A) Interrupt vectoring (B) Cycle stealing
(C) DMA (D) Daisy chain

15. What will be the response of the CPU, on receiving an interrupt from an input/output device?
(A) It hands over the control of address bus and data bus to the interrupting device.
(B) It branches off to the interrupt service routine after completion of the current instruction.
(C) It halts for a predetermined time.
(D) It branches off to the interrupt service routine immediately.

16. What is the bandwidth of memory system that has a latency of 50ns, a pre charge time of 10ns and transfers 2 bytes of data per access?
(A) 60 B/sec (B) 1.67 B/sec
(C) 1.67×10^7 B/sec (D) 3.33×10^7 B/sec

17. A hard disk is connected to a 50MHz processor through a DMA controller. Assume that the initial set-up of a DMA transfer takes 2000 clock cycles for the processor and also assume that the handling of the interrupt at DMA completion requires 1000 clock cycles for the processor. The hard disk has a transfer rate of 4000 K bytes/sec and average block size transferred is 8 K bytes. What fraction of the processor time is consumed by the disk, if the disk is actively transferring 100% of the time?
(A) 1% (B) 1.5%
(C) 2% (D) 3%

18. A device with transfer rate of 20KB/sec is connected to a CPU. Data is transferred byte wise. Let the interrupt overhead is 6 micro seconds. The byte transfer time between the device interface register and CPU or memory is negligible. What is minimum performance gain of operating the device under interrupt mode over operating it under program-controlled mode?
(A) 6 (B) 8
(C) 10 (D) 12

19. A DMA module is transferring characters to main memory from an external device at 76800 bits per second. The processor can fetch instructions at a rate of 2 million instructions per second. How much will the processor be slowed down due to DMA activity? (Express this as a percent of the time from when there is a conflict between DMA and the CPU)
(A) 0.24% (B) 0.48%
(C) 0.96% (D) 0.50%

20. Let us suppose that we want to read 2048 bytes in programmed I/O mode of CPU. The bus width is 32-bits. Each time an interrupt occurs from Hard disk drive and it taken 4 μsec to service it. How much CPU time is required to read 2048 bytes?
(A) 512 msec (B) 768 msec
(C) 1024 msec (D) 2048 msec

Practice Problems 2

Directions for questions 1 to 21: Select the correct alternative from the given choices.

1. Memory which is ultraviolet erasable is
(A) RAM (B) EPROM
(C) PROM (D) EEPROM

2. Memory which is electrically erasable is
(A) EPROM (B) EEPROM
(C) ROM (D) PROM

3. The minimum time delay that is required between the initiation of two successive memory operations is called
(A) Memory access time (B) Transmission time
(C) Seek Time (D) Memory cycle

4. The memory that is programmed at the time of manufacture is
(A) RAM (B) PROM
(C) ROM (D) EEPROM

5. The disadvantage of dynamic RAM over static RAM is
(A) High power consumption
(B) Higher bit density
(C) Need to refresh the capacitor charge every once in two milliseconds.
(D) Variable speed

6. If an error is detected, a part of the memory can be erased in
(A) PROM (B) EPROM
(C) EAROM (D) EROM

7. What are sequences of events in source initiated hand shaking transfer?
(A) Source enable data valid, destination enable data accepted, source disable data valid, destination disable data accepted
(B) Source disable data valid, destination enable data accepted, source disable data valid, destination enable data accepted
(C) Source disable data valid, destination Disable data valid, source enable data valid, destination enable data accepted.
(D) Source disable data valid, destination enable data valid, source enable data valid, destination disable data accepted.

8. Processor needs software interrupt to
(A) return from subroutine
(B) implement co-routines
(C) test the interrupt system of the processor
(D) obtain system services which need execution of privileged instructions

9. A microcomputer has primary memory of 512 KB. what is the exact number of bytes contained in this memory?
(A) 512 × 1000 (B) 512 × 100
(C) 512 × 1024 (D) 512 × 1028

10. The number of address lines required in a microprocessor which has to access 1 K bytes of memory is
(A) 6 (B) 4
(C) 10 (D) 8

11. Software interrupt is
(A) used to stimulate an external device
(B) generated by an external device
(C) Both (A) and (B)
(D) None of these

12. The bus that is used to transfer data from main memory to peripheral devices and vice-versa is
(A) Control bus (B) input bus
(C) output bus (D) DMA bus

13. The bus which is connected between the CPU and the main memory that permits transfer of information between the CPU and main memory is called
(A) memory bus (B) address bus
(C) control bus (D) DMA bus

14. An interrupt in which the external device supplies the interrupt requests as well as its address is called
(A) maskable interrupt
(B) vectored interrupt
(C) designated interrupt
(D) non-maskable interrupt

15. A temporarily ignored interrupt is called
(A) designated interrupt
(B) maskable interrupt
(C) non-maskable interrupt
(D) low priority interrupt

16. Which of the following device is used to connect a peripheral to a bus?
(A) control register
(B) interface
(C) communication protocol
(D) None of these

17. Which of the following is true for the daisy scheme of connecting input/output devices?
(A) It gives non-uniform priority to various devices.
(B) It gives uniform priority to all devices.
(C) It is only useful for connecting slow devices to a processor device.
(D) It requires a separate interrupt pin on the processor for each device.

18. In direct memory access data are directly transferred
(A) from CPU to input/output device and memory
(B) from an input/output device to memory only.
(C) from memory to an input/output device only.
(D) from an input/output device to the memory or vice versa

19. Which one of the following is true for a CPU having a single interrupt request line and a single interrupt grant line?
(A) Vectored interrupt multiple interrupting devices are always possible.
(B) Vectored interrupts are not possible but multiple interrupting devices are possible
(C) Vectored interrupts and multiple interrupting devices are sometimes possible
(D) Vectored interrupt is possible but multiple interrupting devices are not possible

20. In which of the following I/O, there is a single address space for memory locations and I/O devices
(A) Isolated I/O
(B) Memory mapped I/O
(C) DMA
(D) Both (A) and (B)

21. ____ signal used to interrupt processor and to execute service routine that takes an error recovery action.
(A) Strobe (B) Handshaking
(C) Polling (D) Time out

Previous Years' Questions

1. A device with data transfer rate 10 KB/sec is connected to a CPU. Data is transferred byte-wise. Let the interrupt overhead be 4 μsec. The byte transfer time between the device interface register and CPU or memory is negligible. What is the minimum performance gain of operating the device under interrupt mode over operating it under program-controlled mode? **[2005]**
(A) 15 (B) 25
(C) 35 (D) 45

2. A computer handles several interrupt sources of which the following are relevant for this question.
- Interrupt from CPU temperature sensor (raises interrupt if CPU temperature is too high)
- Interrupt from Mouse (raises interrupt if the mouse is moved or a button is pressed)
- Interrupt from Keyboard (raises interrupt when a key is pressed or released)
- Interrupt from Hard Disk (raises interrupt when a disk read is completed)

Which one of these will be handled at the HIGHEST priority? **[2011]**
(A) Interrupt from Hard Disk
(B) Interrupt from Mouse
(C) Interrupt from Keyboard
(D) Interrupt from CPU temperature sensor

The following information pertains to 3 and 4: Consider the following program segment for a hypothetical CPU having three user registers R_1, R_2 and R_3.

Instruction	Operation	Instruction Size (in words)
MOV R_1, 5000;	$R_1 \leftarrow$ Memory [5000]	2
MOV R_2 (R1);	$R_2 \leftarrow$ Memory [(R_1)]	1
ADD R_2, R3;	$R_2 \leftarrow R_2 + R_3$	1
MOV 6000, R_2;	Memory [6000] $\leftarrow R_2$	2
HALT;	Machine halts	1

3. Consider that the memory is byte addressable with size 32 bits, and the program has been loaded starting from memory location 1000 (decimal). If an interrupt occurs while the CPU has been halted after executing the HALT instruction, the return address (in decimal) saved in the stack will be **[2004]**
(A) 1007 (B) 1020
(C) 1024 (D) 1028

4. Let the clock cycles required for various operations be as follows:
Register to/from memory transfer: 3 clock cycles
ADD with both operands in register: 1 clock cycle
Instruction fetch and decode: 2 clock cycles per word
The total number of clock cycles required to execute the program is **[2004]**
(A) 29 (B) 24
(C) 23 (D) 20

5. A CPU generally handles an interrupt by executing an interrupt service routine **[2009]**
(A) As soon as an interrupt is raised.
(B) By checking the interrupt register at the end of fetch cycle.
(C) By checking the interrupt register after finishing the execution of the current instruction.
(D) By checking the interrupt register at fixed time intervals.

6. A hard disk with a transfer rate of 10 Mbytes/second is constantly transferring data to memory using DMA. The processor runs at 6000 MHz, and takes 300 and 900 clock cycles to initiate and complete DMA transfer respectively. If the size of the transfer is 20 Kbytes, what is the percentage of processor time consumed for the transfer operation?
(A) 5.0% (B) 1.0%
(C) 0.5% (D) 0.1%

7. On a non-pipelined sequential processor, a program segment, which is a part of the interrupt service routine, is given to transfer 500 bytes from an I/O device to memory.

Initialize the address register.

Initialize the count to 500

LOOP: Load a byte from device

Store in memory at address given by address register.

Increment the address register

Decrement the count

If count! = 0 go to LOOP

Assume that each statement in this program is equivalent to a machine instruction which takes one clock cycle to execute if it is a non-load/store instruction. The load-store instructions take two clock cycles to execute.

The designer of the system also has an alternate approach of using the DMA controller to implement the same transfer. The DMA controller requires 20 clock cycles for initialization and other overheads. Each DMA transfer cycle takes two clock cycles to transfer one byte of data from the device to the memory.

What is the approximate speedup when the DMA controller based design is used in place of the interrupt driven program based input-output? **[2011]**

(A) 3.4 (B) 4.4
(C) 5.1 (D) 6.7

8. Which of the following statements about synchronous and asynchronous I/O is NOT true? **[2008]**

(A) An ISR is invoked on completion of I/O in synchronous I/O but not in asynchronous I/O
(B) In both synchronous and asynchronous I/O, an ISR (Interrupt Service Routine) is invoked after completion of the I/O
(C) A process making a synchronous I/O call waits until I/O is complete, but a process making an asynchronous I/O call does not wait for completion of the I/O
(D) In the case of synchronous I/O, the process waiting for the completion of I/O is woken up by the ISR that is invoked after the completion of I/O

9. A main memory unit with a capacity of 4 megabytes is built using 1M × 1-bit DRAM chips. Each DRAM chip has 1K rows of cells with 1K cells in each row. The time taken for a single refresh operation is 100 nanoseconds. The time required to perform one refresh operation on all the cells in the memory unit is **[2010]**

(A) 100 nanoseconds
(B) $100 * 2^{10}$ nanoseconds
(C) $100 * 2^{20}$ nanoseconds
(D) $3200 * 2^{20}$ nanoseconds

10. A processor can support a maximum memory of 4GB, where the memory is word - addressable (a word consists of two bytes). The size of the address bus of the processor is atleast ___ bits. **[2016]**

11. The size of the data count register of a DMA controller is 16 bits. The processor needs to transfer a file of 29,154 kilobytes from disk to main memory. The memory is byte addressable. The minimum number of times the DMA controller needs to get the control of the system bus from the processor to transfer the file from the disk to main memory is _____. **[2016]**

12. The following are some events that occur after a device controller issues an interrupt while process *L* is under execution.

(P) The processor pushes the process status of *L* onto the control stack.
(Q) The processor finishes the execution of the current instruction.
(R) The processor executes the interrupt service routine.
(S) The processor pops the process status of *L* from the control stack.
(T) The processor loads the new PC value based on the interrupt.

Which one of the following is the correct order in which the events above occur? **[2018]**

(A) QPTRS (B) PTRSQ
(C) TRPQS (D) QTPRS

13. A 32-bit wide main memory unit with a capacity of 1 GB is built using 256 M × 4-bit DRAM chips. The number of rows of memory cells in the DRAM chip is 2^{14}. The time taken to perform one refresh operation is 50 nanoseconds. The refresh period is 2 milliseconds. The percentage (rounded to the closest integer) of the time available for performing the memory read write operations in the main memory unit is _____. **[2018]**

Answer Keys

Exercises

Practice Problems 1

1. A	**2.** C	**3.** D	**4.** B	**5.** B	**6.** C	**7.** B	**8.** C	**9.** C	**10.** A
11. C	**12.** D	**13.** A	**14.** D	**15.** B	**16.** D	**17.** D	**18.** B	**19.** B	**20.** D

Practice Problems 2

1. B	**2.** B	**3.** A	**4.** C	**5.** A	**6.** C	**7.** B	**8.** D	**9.** C	**10.** C
11. A	**12.** D	**13.** A	**14.** B	**15.** B	**16.** B	**17.** A	**18.** D	**19.** B	**20.** B
21. D									

Previous Years' Questions

1. B	**2.** D	**3.** D	**4.** B	**5.** C	**6.** D	**7.** A	**8.** A	**9.** D	**10.** 31
11. 456	**12.** A	**13.** 59 to 60							

Chapter 4

Instruction Pipelining

LEARNING OBJECTIVES

- *Flynn's classification*
- *Pipelining*
- *Six-stages of pipelining*
- *Pipeline performance*
- *Pipeline hazards*
- *Structural hazards*
- *Data hazards*
- *Control hazards*
- *Conditional branch*
- *Dealing with branches*

Flynn's Classification

In parallel processing, the system is able to perform concurrent data processing to achieve faster execution time. A classification introduced by M.J. Flynn considers the organization of a computer system by the number of instructions and data items that are manipulated simultaneously. According to this classification, there are four major groups of computers:

1. **Single instruction stream, single data stream (SISD):**
   - Single computer containing a control unit, a processor unit and a memory unit
   - Instructions executed sequentially; parallel processing may be achieved by multiple functional units or by pipeline processing.
2. **Single instruction stream, Multiple data stream (SIMD):**
   - Include many processing units under the supervision of a common control unit.
   - All processors receive the same instruction from the control unit but operate on different items of data.
3. **Multiple instruction stream, Single data stream (MISD):**
   - No practical system has been constructed.
4. **Multiple instruction stream, Multiple data stream (MIMD):**
   - Capable of processing several programs at the same time.

One type of parallel processing that does not fit Flynn's classification is pipelining.

Pipelining

Pipelining is a technique of decomposing a sequential process into sub operations, with each subprocess being executed in special dedicated segment that operates with all other segments.

In pipelining, new inputs are accepted at one end before previously accepted inputs appear as outputs at the other end.

Two-stage Pipeline

As a simple approach, consider subdividing instruction processing into two stages:

1. Fetch instruction
2. Execute instruction
   - There are times during the execution of an instruction when main memory is not being accessed. This time can be used to fetch the next instruction in parallel with the execution of the current one. This is called instruction prefetch or fetch overlap.
   - This process will speed up instruction execution. If the fetch and execute stages were of equal duration, the instruction cycle time would be halved.
   - But there are some problems in this technique:
     (i) The execution time is generally longer than the fetch time.
     (ii) A conditional branch instruction makes the address of the next instruction to be fetched unknown.
   - These two factors reduce the potential effectiveness of the two-stage pipeline, but some speed up occurs. To gain further speed up, the pipeline must have more stage(s).

Six-stage Pipeline

Let the six stages be

F: Fetch Instruction
D: Decode Instruction
C: Calculate Operand Address
O: Operand Fetch

E: Execute instruction
W: Write operand

Then the time line diagram for seven instructions is shown below:

Clock cycle	1	2	3	4	5	6	7	8	9	10	11	12
Instruction *i*	F	D	C	O	E	W						
i+1		F	D	C	O	E	W					
i+2			F	D	C	O	E	W				
i+3				F	D	C	O	E	W			
i+4					F	D	C	O	E	W		
i+5						F	D	C	O	E	W	
i+6							F	D	C	O	E	W

Execution Time for the seven instructions with pipelining $= \left(\frac{t_{ex}}{6}\right) \times 12 = 2 * t_{ex}$. Where t_{ex} in the execution time required for each instruction.

- A deeper pipeline means that there are more stages in the pipeline. This generally means that the processor's frequency can be increased as the cycle time is lowered. This happen because there are fewer components in each stage of the pipeline, so the propagation delay is decreased for the overall stage.
- An instruction pipeline is said to be fully pipelined if it can accept a new instruction in every clock cycle.
- A pipeline that is not fully pipelined has wait cycle that delays the progress of the pipeline.

Advantages of pipelining:
- The cycle time of the processor is reduced, thus increasing instruction issue rate in most cases.
- Some combinational circuits such as adders or multipliers can be made faster by adding more circuitry. If pipelining is used instead it can save circuitry versus a more complex combinational circuit.

Limitations of pipelining:
1. If the stages are not of equal duration, there will be some waiting involved at various stages.
2. Conditional branch instruction may invalidate several instruction fetches.
3. The contents of one stage may depend on the contents of other stages of previous instructions, which is still in pipeline.

Pipeline Performance

The cycle time τ of an instruction pipeline is the time needed to advance a set of instructions one stage through the pipeline.

$$\text{Cycle time} = \max[\tau_i] + d = \tau_m + d,\ 1 \le i \le K$$

where τ_i = Time delay of the circuitry in the i^{th} stage of the pipeline.

τ_m = maximum stage delay.
K = number of stages in instruction pipeline
d = time delay of a latch, needed to advance signals and data from one stage to the next.

Suppose that n instructions are processed without any branches. Let $T_{k,n}$ be the total time required for a pipeline with K stages to execute n instructions. Then

$$T_{k,n} = [K+(n-1)]\tau$$

Example 1: Let $n = 7, K = 6, \tau = 1$. Then $T_{k,n} = [6 + (7 - 1)] \times 1 = 12$ cycles.

Now consider a processor with equivalent functions but no pipeline and assume that the instruction cycle time is $k\tau$. The speed up factor for the instruction pipeline compared to execution without the pipeline is defined as

$$S_k = \frac{T_{1,n}}{T_{k,n}} = \frac{nk\tau}{[(k+(n-1)]\tau} = \frac{nk}{k+(n-1)}$$

Note: Larger the number of stages, greater the potential for speed up. But practically, the potential gains of additional pipeline stages are countered by increase in cost, delays between stages, and the fact that braches will be encountered requiring the flushing of the pipeline.

Arithmetic pipeline:
- Pipeline arithmetic units are usually found in very high-speed computers.
- These are used to implement floating point operations, multiplication of fixed point numbers and similar computations encountered in scientific problems.

Pipeline Hazards

- Pipeline hazards are situations that prevent the next instruction in the instruction stream from executing during its designated clock cycle. The instruction is said to be stalled. When an instruction is stalled, all instructions later in the pipeline than the stalled instructions are also stalled. Instructions earlier than the stalled one can continue. No new instructions are fetched during the stall.

Note: Keeping a pipeline at its maximal rate is prevented by pipeline hazard.

Different types of Hazards:

1. Structural Hazards
2. Data Hazards
3. Control Hazards

Structural Hazards

Structural hazards occur when a certain resource is requested by more than one instruction at the same time.

Example 2: Instruction MVI *B*, *X* fetches in the *O* stage operand *X* from memory. The memory does not accept another access during that cycle.

Clock cycle	1	2	3	4	5	6	7	8	9	10	11
MVI *B*, *X*	F	D	C	O	E	W					
Instruction $i+1$		F	D	C	O	E	W				
$i+2$			F	D	C	O	E	W			
$i+3$				stall	F	D	C	O	E	W	
$i+4$						F	D	C	O	E	W

Penalty: 1 cycle

Certain resources are duplicated in order to avoid structural hazards (ALU, floating-point unit) can be pipelined themselves in order to support several instructions at a time. A classical way to avoid hazards at memory access is by providing separate data and instruction caches.

Note: Structural hazards are due to resource conflict.

Data Hazards

In a pipeline execution of two instructions I_1 and I_2 a certain stage of the pipeline I_2 needs the result produced by I_1, but this result has not yet been generated, then we have a data hazard.

Example 3: I_1: ADD R_3, R_2 $\quad R_3 \leftarrow R_3 + R_2$

I_2: MUL R_1, R_3 $\quad R_1 \leftarrow R_1 * R_3$

Clock cycle	1	2	3	4	5	6	7	8	9	10	11	12
ADD R_3, R_2	F	D	C	O	E	W						
MUL R_1, R_3		F	D	C	stall	stall	O	E	W			
Instruction $i+2$			F	D			C	O	E	W		

Penalty: 2 cycles Before executing the O stage (operand fetch stage), the MUL instruction is stalled until the ADD instruction has written the result into R_3.

Data dependencies

Data dependency exists between two instructions if the data data used by an instruction depends on the data created by other instructions.

Two type of dependencies exist between instructions:

1. True data dependency
2. Name dependencies
   (i) Anti-dependency
   (ii) Output dependency

True data dependency

- This is also called as Read-After-Write Hazard (RAW).
- This type of dependency occurs when the value produced by an instruction is required by a subsequent instruction.
- This is also known as a flow dependency because dependency is due to flow of data in a program.

Example: ADD R_3, R_2, R_1; $R_3 \leftarrow R_2 + R_1$

SUB R_4, R_3, 1; $R_4 \leftarrow R_3 - 1$

- Here R_3 is read before it is written by 'ADD' instruction.
- In RAW hazard $(i+1)^{st}$ instruction tries to read a source before it is written by 'i^{th}' instruction. So $(i+1)^{st}$ instruction incorrectly gets the old value.
- This kind of hazard can be reduced by using forwarding (or Bypassing).

Name dependencies

1. **Anti-dependency:**
   - This is also called as Write-After-Read hazard
   - This kind of dependency occurs when an instruction writes to a location which has been read by a previous instruction.
   - Here $(i+1)^{st}$ instruction tries to write an operand before it is read by i^{th} instruction. So i^{th} instruction incorrectly gets the new value.

Example: I_1: ADD R_3, R_2, R_1 ;$R_3 \leftarrow R_2 + R_1$

I_2: SUB R_2, R_5, 1; $R_2 \leftarrow R_5 - 1$

I_2 must not produce its result in R_2 before I_1 read R_2, otherwise I_1 would use the value produced by I_2 rather than the previous value of R_2.

2. **Output dependency:**
   - This is also called as Write - After - Write (WAW) hazard.
   - This dependency occurs when a location is written by two instructions.
   - i.e., $(i+1)$th instruction tries to write an operand before it is written by i^{th} instruction.

The writes end up being performed in the wrong order.

Example: I_1: ADD R_3, R_2, R_1;$R_3 \leftarrow R_2 + R_1$

I_2: SUB R_2, R_3, 1; $R_2 \leftarrow R_3 - 1$

I_3: R_3, R_2, R_5; $R_3 \leftarrow R_2 + R_5$

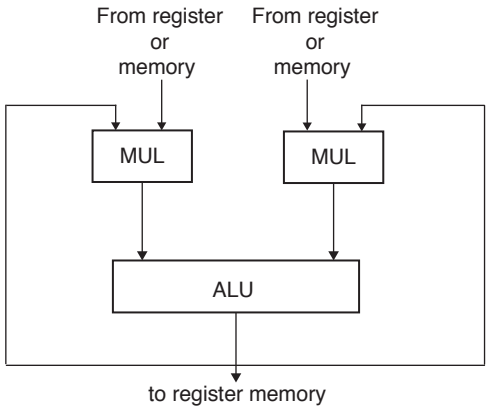
There is a possibility of WAW hazard between I_1 and I_3.

Handling data dependency There are ways to handle date dependency.

1. Hardware interlocks
2. Operand forwarding
3. Delayed load

1. **Hardware interlocks:**
   - To avoid data dependency, insert hardware interlock.
   - An interlock is a circuit that detects instructions whose source operands are destinations of instructions farther up in the pipeline.
2. **Operand forwarding:**
   - Uses special hardware to detect a conflict and then avoid it by routing the data through special paths between pipeline segments.

- Some of the penalty produced by data hazards can be avoided using a technique called forwarding (Bypassing).
- The ALU result is always fed back to the ALU input. If the hardware detects that the value needed for the current operation is the one produced by the previous operation (but which has not yet been written back). It selects the forwarded result as the ALU input, instead of the value read from register or memory.



Clock cycle	1	2	3	4	5	6	7	8
ADD R_3, R_2	F	D	C	O	E	W		
MUL R_1, R_3		F	D	C	stall	O	E	W

Penalty: 1 cycle After the E stage of the MUL instruction the result is available by forwarding. Therefore the penalty is reduced to one cycle.

Delayed Load: Here the compiler of a computer will detect the data conflicts and reorder the instructions as necessary to delay the loading of the conflicting data by inserting no-operation instruction.

Control Hazards

Control hazards are produced by Branch Instructions.

Unconditional branch

- Jump loop
- Loop

Clock cycle	1	2	3	4	5	6	7	8	9	10	11
Loop	F	D	C	O	E	W					
Loop		stall	stall	stall	F	D	C	O	E	W	
Loop + 1						F	D	C	O	E	W

Penalty: 3 cycles

- The instruction following the branch is fetched before the D stage is finished in 2nd clock. It is not known that a branch is executed. Later the fetched instruction is discarded.
- After the O stage of the branch instruction the address of the target is known and it can be fetched.

Conditional branch

Example: ADD B; $A \leftarrow A + B$
JZ Loop
Loop: If condition satisfies and branch is taken:

Clock cycle	1	2	3	4	5	6	7	8	9	10	11	12
ADD *B*	F	D	C	O	E	W						
JZ Loop		F	D	C	O	E	W					
Loop			stall	stall	stall	F	D	C	O	E	W	

Penalty:3 cycles

At this moment both the condition (set by ADD) and the target address are known.

If condition not satisfied and branch not taken:

Clock cycle	1	2	3	4	5	6	7	8	9	10	11	12
ADD *B*	F	D	C	O	E	W						
JZ Loop		F	D	C	O	E	W					
Instruction *i* + 1			F	stall	stall	D	C	O	E	W		

Penalty:2 cycles

At this moment the condition is known and instruction i + 1 can go on.

- With conditional branch, we have a penalty even if the branch has not been taken. This is because we have to wait until the branch condition is available.

Dealing with branches

One of the major problems in designing an instruction pipeline is the occurrence of branch instructions. A variety of approaches have been taken for dealing with braches

1. Multiple streams
2. Prefetch branch target
3. Branch target buffer
4. Loop buffer
5. Branch prediction
6. Delayed branch

Multiple streams A branch instruction may cause to choose one of two instructions to fetch next, then allow the pipeline to fetch both instructions, making use of streams.

There are two problems with this approach:

1. With multiple pipelines there are contention delays for access to the registers and to memory.
2. Additional branch instructions may enter the pipeline before the original branch decision is resolved.

Prefetch branch target When a conditional branch is recognized, the target of the branches is prefetched, in addition to the instruction following the branch. This target is then saved until the branch instruction is executed.

Branch targets buffer (BTB) BTB is an associative memory included in the fetch segment of the pipeline. Each entry in the BTB consists of the address of a previously executed branch instruction and target instructions for that branch.

It also stores the next few instructions after the branch target instruction.

When the pipeline decodes a branch instruction, it searches the associative memory BTB for the address of the instruction. If it is in BTB, the instruction is available directly and prefetch continues from the new path. If the instruction is not in BTB, the pipeline shifts to a new instruction stream and stores the target instruction in the BTB.

Advantage: Branch instructions that occurred previously are readily available in the pipeline without interruption.

Loop Buffer A loop buffer is a small, very high speed memory maintained by the instruction fetch stage of the pipeline and containing the n most recently fetched instructions in sequence. If a branch is to be taken, the hardware first checks whether the branch target is within the buffer. If so, the next instruction is fetched from the buffer. The Advantages of loop buffer are

1. Loop buffer will contain some instructions sequentially ahead of the current instruction fetch address. Thus instructions fetched in sequence will be available without the usual memory access time.
2. If the branch occurs to a target just a few locations ahead of the address of the branch instruction, the target will already be in the buffer.
3. This strategy is well suited in dealing with loops.

Branch prediction Various techniques can be used to predict whether a branch will be taken or not. The common techniques are

Static { 1. Predict never taken Static
2. Predict always taken
3. Predict by opcode

Dynamic { 4. Taken/not taken switch
5. Brach history table

- The first two approaches are static, i.e., no dependency on execution history. Here always assume that the branch will not be taken and continue to fetch instructions in sequence, or always assume that the branch will be taken and always fetch from the branch target.
- The third approach is also static. Takes the decision based on the opcode of the branch instruction in a program.
- Dynamic branch strategies attempt to improve the accuracy of prediction by recording the history of conditional branch instructions in a program.

(a) Taken/not taken switch:

- Use two bits to record the result of the last two instances of the execution of the associated instruction or record a state in some other fashion.

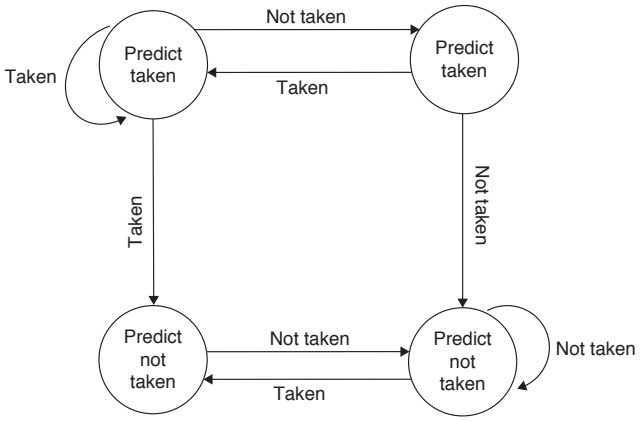


Figure 1 Branch prediction state diagram

- As long as each succeeding conditional branch instruction that is encountered is taken, the decision process predicts that the next branch will be taken.
- If a single prediction is wrong, the algorithm continues to predict that the next branch is taken.
- Only if two successive branches are not taken does the algorithm shift to not taken branch.

Drawback: If the decision is made to take the branch, the target instruction cannot be fetched until the target address, which is an operand in the conditional branch instruction is decoded.

(b) Branch history table: It is a small cache memory associated with the instruction fetch stage of the pipeline.

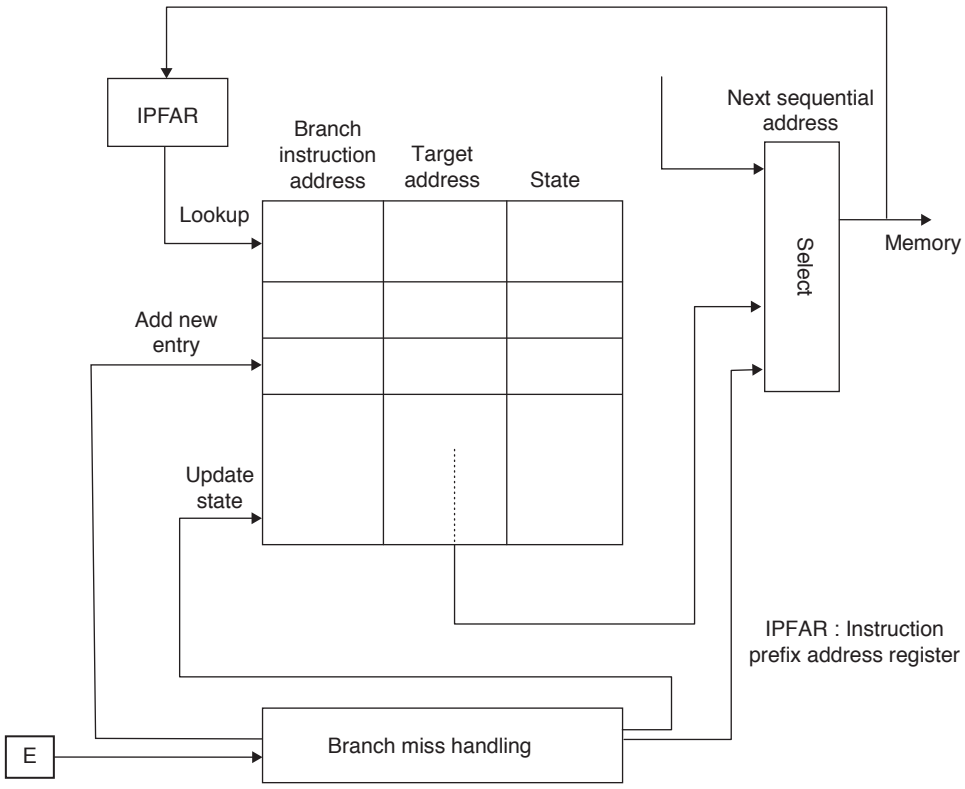


Figure 2 Branch history table

- Each entry in the table consist of three elements:
  1. The address of branch instruction.
  2. Some number of history bits that record the state of use of that instruction.
  3. Information about target instruction.

Delayed branch A complier detects the branch instructions and rearranges the machine language code sequence by inserting useful instructions that keep a pipeline operating without interruptions.

Exercises

Practice Problems I

Directions for questions 1 to 21: Select the correct alternative from the given choices.

Common data for questions 1 and 2: An unpipelined processor with eight number cycle time and pipeline batches with 1 ns latency is given.

1. Find the cycle times of pipelined versions of the processor with 2, 4, 8 and 16 stages if the Data path logic is evenly divided among the pipeline stages.
(A) 5, 3, 2, 1.5 (B) 4, 2, 1, 0.5
(C) 8, 4, 2, 1 (D) 10, 6, 4, 3

2. What is the latency of each of the pipelined versions of the processor?
(A) 4, 2, 1, 0.5 (B) 10, 6, 4, 3
(C) 5, 3, 2, 1.5 (D) 10, 12, 16, 24

3. A 4-stage pipeline has the stage delays as 110, 120, 130, and 140 nanoseconds respectively. Registers that are used between the stages have a delay of 2 nanoseconds each. Assuming constant clocking rate. Find the total time taken to process 1000 instructions on this pipeline.
(A) 7.1 ms (B) 14.24 ms
(C) 28 ms (D) 2000 ms

4. Consider a pipelined processor with the following four stages:
IF: instruction fetch
ID: Instruction decode
EX: Execute
WB: Write back
The IF, ID and WB stages takes one clock cycle each to complete the operation. The number of clock cycles for EX stage depends on the instruction; for I_1 and I_3 one clock cycle is needed and for I_2 three clock cycles are needed. Find the number of clock cycles taken to complete the following sequence of instructions?

I_1:	ADD R_0,	R_1,	R_2	$R_0 \leftarrow R_1 + R_2$
I_2:	MUL R_2,	R_3,	R_4	$R_2 \leftarrow R_3 \times R_4$
I_3:	SUB R_4,	R_5,	R_6	$R_4 \leftarrow R_5 - R_6$

(A) 7 (B) 8
(C) 6 (D) 9

5. A CPU has five stage pipelines and runs at 1 GHz frequency. Instruction fetch happens in the first stage of the pipeline. A conditional branch instruction computes the target address and evaluates the condition in the third stage of the pipeline. The processor stops fetching new instructions following a conditional branch until the branch outcome is known. A program executes 10^9 instructions. Out of which 10% are conditional branches. If each instruction takes one cycle to complete on average then find the total execution time of the program?
(A) 1 sec (B) 1.2 sec
(C) 1.4 sec (D) 1.8 sec

6. Consider a four stage pipeline processor, number of cycles needed by the four instructions I_1, I_2, I_3 and I_4 in stages S_1, S_2, S_3 and S_4 are shown below:

	S_1	S_2	S_3	S_4
I_1	2	1	1	1
I_2	1	3	2	2
I_3	2	1	1	3
I_4	1	2	2	2

What is the number of cycles needed to execute the instructions in the order:
$I_1 : I_2 : I_3 : I_4$
(A) 8 (B) 12
(C) 14 (D) 15

7. A non-pipelined system takes 50 ns to process a task; the same task can be processed in a six-segment pipeline with a clock cycle of 10 ns. Speedup ratio of 100 tasks for pipeline is
(A) 1.62 (B) 3.21
(C) 4.76 (D) 8.21

8. Consider a pipelined processor with the following four stages:
IF: Instruction fetch
ID: Instruction decode
EX: Execute
WB: Write back

The IF, ID and WB stages takes one clock cycle each to complete the operation. The number of clock cycles for EX stage depends on the instruction; for I_1 and I_3 one clock cycle is needed and for I_2 three clock cycles are needed. The number of clock cycles taken to complete the following sequence of instructions is

I_1:	ADD R_0, R_1, R_2,	$R_0 \leftarrow R_1 + R_2$
I_2:	MUL R_2, R_0, R_4,	$R_2 \leftarrow R_0 \times R_4$
I_3:	SUB R_4, R_5, R_2,	$R_4 \leftarrow R_5 - R_2$

(A) 7 (B) 8
(C) 9 (D) 10

9. Following are the sequence of stages in a pipeline CPU:
(1) IF: Instruction fetch from instruction memory
(2) RD: Instruction decode and register read
(3) EX: Execute ALU operation for data and address computation
(4) MA: Data memory access, for write access, the register read at RD stage is used.
(5) WB: Register write back

Consider the following sequence of instructions:
LOAD R_1, M[loc]
ADD R_1, R_1, R_1
ADD R_2, R_1, R_2
Let each stage take one clock cycle.

What is the number of clock cycles taken to complete the above sequence of instructions starting from the fetch of first instruction?
(A) 18 (B) 15
(C) 13 (D) 10

10. Which of the following can cause a hazard for a pipelined CPU with a single ALU?
(i) The $(j + 1)^{st}$ instruction uses the result of the j^{th} instruction as an operand.
(ii) The j^{th} and $(j + 1)^{st}$ instructions require the ALU at the same time.
(iii) The execution of a conditional jump instruction.
(iv) The execution of non-conditional jump instruction.
(A) (i) and (ii) (B) (ii) and (iii)
(C) (i), (ii) and (iii) (D) (i), (ii), (iii) and (iv)

11. Given an unpipelined processor with a 10 ns cycle time and pipeline latches with 0.5 ns latency, how many stages of pipelining are required to achieve cycle time of 2 ns?
(A) 5.5 (B) 6.67
(C) 7 (D) 6

12. In a 4-stage pipeline,
IF – instruction fetches
ID – instruction decode and fetch operands
EX – Execute
WB – write back
ADD, SUB take one clock cycle, MUL take three clock cycles. Then for
ADD R_2, R_1, R_0 $R_2 \leftarrow R_1 + R_0$
MUL R_4, R_3, R_2 $R_4 \leftarrow R_3 * R_2$
SUB R_6, R_5, R_4 $R_6 \leftarrow R_5 - R_4$
Number of clock cycles required using operand forwarding technique are
(A) 8 (B) 12
(C) 10 (D) 14

13. Consider an instruction sequence of length 'n' that is streaming through a K-stage instructions pipeline. Let P be the probability of encountering a conditional or

unconditional branch instruction and let q be the probability that execution of a branch instruction I_B causes a jump to a non-consecutive address. Assume that each such jump requires the pipeline to be cleared, destroying all ongoing instruction processing, when I_B emerges from the last stage. Also assume that T is the cycle time. Then which of the following expression correctly specifies the time required for this pipeline?

(A) $pqnk\tau + (1 - pq)\,[K + (n - 1)]\tau$
(B) $(1 - pq)\,[k + (n - 1)]\tau + pqn\tau$
(C) $pqnk\tau + (1 - pq)\,n[k + (n - 1)]\tau$
(D) $pqn + (1 - pq)n\,[k + (n - 1)]\tau$

14. If T_m is maximum stage delay of an m-stage pipeline with time delay of the latch is d then cycle time is

(A) T_m/d (B) $T_m + d$
(C) $2T_m + d$ (D) $T_m \times d$

15. Pipelining is a general technique for increasing processor ____ without requiring large amounts of extra hardware.

(A) turnaround time (B) waiting time
(C) latency (D) throughput

16. A 4-stage instruction pipeline executes a 100 instruction program. The probability of occurrence of a conditional or unconditional branch is 0.4 and the probability of execution of a branch instruction I_B causing a jump to a non-consecutive address is 0.1. Then the speed up factor for the instruction pipeline compared to execution without pipeline is

(A) 2.14 (B) 6.23
(C) 3.21 (D) 3.48

17. A non-pipelined processor has a clock rate of 2.5GHz and an average cycles per instruction of 4. An upgrade to the processor introduces a five stage pipeline. However, due to internal pipeline delays, such as latch delay, the clock rate of the new processor has to be reduced to 2GHz. What is the MIPS rate for each of these processors respectively.

(A) 625, 400 MIPS (B) 625, 2000 MIPS
(C) 3125, 2000 MIPS (D) 3125, 400 MIPS

18. Consider the following sequence of instructions:

I_1: MUL R_1, R_2 $R_1 \leftarrow R_1 * R_2$
I_2: SUB $R_3, 1$ $R_3 \leftarrow R_3 - 1$
I_3: ADD R_3, R_4 $R_3 \leftarrow R_3 + R_4$
I_4: BEZ Target Branch if zero
I_5: MOVE R_3,10 $R_3 \leftarrow 10$
⋮

Target:

Which of the following instruction will be placed in delayed slot to reduce penalty in a 6-stage pipeline? (Assume that the branch outcome will be known during 5th stage)

(A) I_1 (B) I_2
(C) I_3 (D) I_5

19. Consider the following sequence of instructions:

ADD R_1, R_2 $R_1 \leftarrow R_1 + R_2$
BEZ Target Branch if Zero
MUL R_3, R_4 $R_3 \leftarrow R_3 * R_4$
MOVE R_1,10 $R_1 \leftarrow 10$
⋮

Target:

Assume that this program executed on a 6-stage pipelined processor and each stage required 1 clock cycle.

Let us suppose that "branch not taken" Prediction is used but the prediction is not fulfilled, then the penalty will be (branch outcome is known at 5th stage)

(A) 1 clock cycle (B) 2 clock cycles
(C) 3 clock cycles (D) 4 clock cycles

20. Suppose 40% of the instructions are loads and half the time they are followed by instruction that depends on value loaded. If this hazard causes single cycle delay, how must faster is ideal pipelined machine (CPI = 1) than real one? (Ignore other stalls)

(A) 1 time (B) 1.2 times
(C) 1.5 times (D) 1.15 times

21. Assume that a pipelined processor has three categories of instructions: Branch, load/store, other. If it is a branch instruction it will take 3 clock cycles, if it is a load/store instruction it will take 4 clock cycles and all other instructions require 6 clock cycles. A program consisting of 10% Branch instructions, 10% of load/store instructions is executed on this processor. Then the number of clock cycles required for the execution of the program is

(A) 2.45 (B) 3.61
(C) 4.66 (D) 5.5

Practice Problems 2

Directions for questions 1 to 20: Select the correct alternative from the given choices.

1. The time required for the five functional units, which operated in each of the five cycles are 10 ns, 7 ns, 10 ns, 10 ns and 8 ns. Assume that pipelining add 1 ns of overhead. The speed up of pipeline compared to unpipeline is

(A) 4.5 times (B) 1.1 times
(C) 4.1 times (D) 2.4 times

2. Which of the following is a technique of decomposing a sequential process into sub operations with each subprocess being executed in a special dedicated segment that operates concurrently with each other?

(A) Straight line sequencing
(B) Random sequencing
(C) Pipelining
(D) Serial execution

3. Which of the following statements is incorrect?
(A) Latency is the number of time units between two initiations in a pipelined architecture.
(B) If initiations are of different but fixed reservation tables, the architecture is known as static pipelined configuration.
(C) A collision in a pipelined architecture is an attempt by two different initiations to use the same stage at the same time.
(D) None of the above

4. Which of the following technique is used in a pipelined processor, when there is a conditional branch?
(A) Loop butter (B) Branch prediction
(C) Delayed Branch (D) All of the above

5. Which of the following cases, leads to a pipelined computer architecture?
(A) The evaluation of each basic function is relatively independent of the previous one.
(B) The sub-functions are closely related to each other.
(C) The evaluation of each sub function requires approximately the same sequence.
(D) All of the above

6. The performance of a pipelined processors is degraded if
(A) the pipeline stages have different delays.
(B) consecutive instructions are dependent on each other.
(C) the pipeline stages share hardware resources.
(D) All of the above

7. The following is a limit on how much the performance of a processor can be improved using pipelining:
(A) the number of pipeline stages
(B) data dependencies
(C) branch delays
(D) All of the above

8. A pipeline processor consists of a sequence of 'm' data processing circuits called ____, which collectively perform a single operation on a stream of data operands passing through them.
(A) stages (B) pipelines
(C) latches (D) None of the above

9. A five-stage pipelined CPU has the following sequence of stages:
IF – Instruction fetch from memory
RD – Decode instruction
EX – Execute
MA – Data memory access
WB – Register write back

Consider the following instruction sequence:
I_1: Load R_0 $R_0 \Leftarrow M$
I_2: ADD R_1, $R_1 R_1 \Leftarrow R_1 + R_1$
I_3: SUB R_2, $R_3 R_2 \Leftarrow R_2 - R_3$

Each stage takes one clock cycle.
Number of clock cycles to execute the program is
(A) 8 (B) 10
(C) 7 (D) 15

Common data for questions 10 and 11: Given an unpipelined processor with a 10 number cycle time and pipeline latches with 0.5 ns latency.

10. Which are the cycle times of pipelined versions of the processors with 2, 4, 8 and 16 stages if the data path logic is evenly divided among the pipeline stages?
(A) 5.0, 3.0, 1.5, 1.0
(B) 5.5, 3.0, 1.75, 1.125
(C) 4.0, 5.0, 6.0, 7.0
(D) None of the above

11. What is the latency of each of the pipelined versions of the processor with 2, 4, 8 and 16 stages?
(A) 10, 11, 12, 14 ns
(B) 10, 10, 11, 11 ns
(C) 11, 12, 14, 18 ns
(D) None of the above

12. Assume an unpipelined processor has a 1 ns clock cycle and it uses 5 cycles for ALU operations and branches. And 6 clock cycles for memory operations. A program has 40%, 30%, and 20% of ALU operations, branch instructions and memory operations respectively. If we are using pipelining it adds 0.2 ns overhead. Then what is the speedup of pipelining compared to unpipelined processor?
(A) 1.2 (B) 3.91
(C) 4.7 (D) 2.5

13. Consider a five-stage pipeline processor in which each instructions on an average has 2 clock cycle stalls. Then the speed up of this pipelined processor compared to an unpipelined processor is
(A) 2.5 (B) 1.67
(C) 0.4 (D) 5

14. Pipelining strategy is called to implement
(A) instruction execution
(B) instruction prefetch
(C) instruction decoding
(D) instruction manipulation

15. If an Instruction 'j' tries to read a source operand before instruction 'i' writes it. Then it is a ____ type of hazard.
(A) WAR (B) RAW
(C) WAW (D) None of these

16. What is the average instruction processing time of a five-stage instruction pipeline for 32 instructions if conditional branch instructions occur as follows: I_2, I_5, I_7, I_{25}, I_{27}.
(A) 1.97 (B) 1.67
(C) 1.75 (D) 1.25

17. Consider the execution of 1000 instructions on a five-stage pipeline machine. Then the speed-up due to the use of pipelining given that the probability of an instruction being a branch is 0.2.
(A) 1.77 (B) 2.6
(C) 2.77 (D) 3.2

18. If an instructions following a branch (taken or not taken) have a dependency on the branch and cannot be executed until the branch is executed, then the dependency is
(A) True data dependency
(B) Procedural dependency
(C) Resource conflict
(D) Output dependency

19. 'A two-stage instruction pipeline unlikely to cut the instruction cycle time in half, compared with the use of no pipeline.' The statement is
(A) Always true (B) Always False
(C) Can't predict (D) Some times true

20. Write after read dependency is also known as
(A) True dependency (B) Anti-dependency
(C) Output dependency (D) Inverse dependency

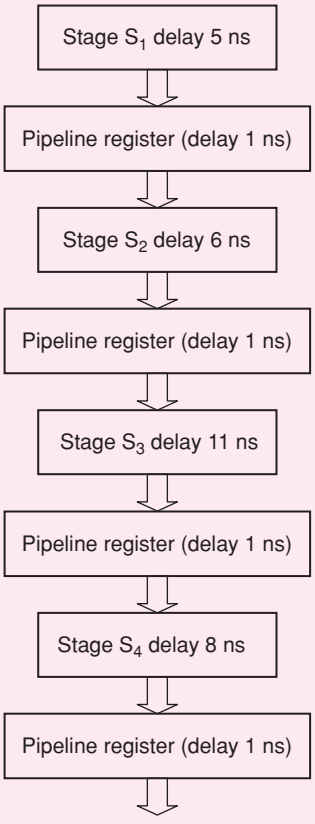
Previous Years' Questions

1. Consider a 6-stage instruction pipeline, where all stages are perfectly balanced. Assume that there is no cycle time overhead of pipelining. When an application is executing on this 6-stage pipeline, the speedup achieved with respect to non-pipelined execution if 25% of the instructions incur 2 pipeline stall cycles is _____. **[2014]**

2. Consider the following processors (ns stands for nano-seconds). Assume that the pipeline registers have zero latency.
P1: Four-stage pipeline with stage latencies 1 ns, 2 ns, 2 ns, 1 ns.
P2: Four-stage pipeline with stage latencies 1 ns, 1.5 ns, 1.5 ns, 1.5 ns.
P3: Five-stage pipeline with stage latencies 0.5 ns, 1 ns, 1 ns, 0.6 ns, 1 ns.
P4: Five-stage pipeline with stage latencies 0.5 ns, 0.5 ns, 1 ns, 1 ns, 1.1 ns.
Which processor has the highest peak clock frequency? **[2014]**
(A) P1 (B) P2
(C) P3 (D) P4

3. An instruction pipeline has five stages, namely, instruction fetch (IF), instruction decode and register fetch (ID/RF), instruction execution (EX), memory access (MEM), and register write back (WB) with stage latencies 1 ns, 2.2 ns, 2 ns, 1 ns, and 0.75 ns, respectively (ns stands for nano seconds). To gain in terms of frequency, the designers have decided to split the ID/RF stage into three stages (ID, RF1, RF2) each of latency 2.2/3 ns. Also, the EX stage is split into two stages (EX1, EX2) each of latency 1 ns. The new design has a total of eight pipeline stages. A program has 20% branch instructions which execute in the EX stage and produce the next instruction pointer at the end of the EX stage in the old design and at the end of the EX2 stage in the new design. The IF stage stalls after fetching a branch instruction until the next instruction pointer is computed . All instructions other than the branch instruction have an average CPI of one in both the designs. The execution times of this program on the old and the new design are P and Q nanoseconds, respectively. The value of P/Q is _____. **[2014]**

4. Consider an instruction pipeline with four stages (S_1, S_2, S_3 and S_4) each with combinational circuit only. The pipeline registers are required between each stage and at the end of the last stage. Delays for the stages and for the pipeline registers are as given in the figure.



What is the approximate speed up of the pipeline in steady state under ideal conditions when compared to the corresponding non-pipeline implementation? **[2011]**

(A) 4.0 (B) 2.5
(C) 1.1 (D) 3.0

5. Register renaming is done in pipelined processors **[2012]**

(A) as an alternative to register allocation at compile time
(B) for efficient access to function parameters and local variables
(C) to handle certain kinds of hazards
(D) as part of address translation

6. Consider an instruction pipeline with five stages without any branch prediction: Fetch Instruction (FI), Decode Instruction (DI), Fetch Operand (FO), Execute Instruction (EI) and Write Operand (WO). The stage delays for FI, DI, FO, EI and WO are 5 ns, 7 ns, 10 ns, 8 ns and 6 ns, respectively. There are intermediate storage buffers after each stage and the delay of each buffer is 1 ns. A program consisting of 12 instructions $I_1, I_2, I_3, \ldots, I_{12}$ is executed in this pipelined processor. Instruction I_4 is the only branch instruction and its branch target is I_9. If the branch is taken during the execution of this program, the time (in ns) needed to complete the program is **[2013]**

(A) 132 (B) 165
(C) 176 (D) 328

Common data for Questions 7 and 8: Delayed branching can help in the handling of control hazards

7. For all delayed conditional branch instructions, irrespective of whether the condition evaluates to true or false **[2008]**

(A) The instruction following the conditional branch instruction in memory is executed
(B) The first instruction in the fall through path is executed
(C) The first instruction in the taken path is executed
(D) The branch takes longer to execute than any other instruction

8. The following code is to run on a pipelined processor with one branch delay slot:

I_1: ADD $R_2 \leftarrow R_7 + R_8$
I_2: SUB $R_4 \leftarrow R_5 - R_6$
I_3: ADD $R_1 \leftarrow R_2 + R_3$
I_4: STORE Memory $[R_4] \leftarrow R_1$
BRANCH to Label if $R_1 = 0$

Which of the instructions I_1, I_2, I_3 or I_4 can legitimately occupy the delay slot without any other program modification? **[2008]**

(A) I_1 (B) I_2
(C) I_3 (D) I_4

9. A 5 stage pipelined CPU has the following sequence of stages:

IF - Instruction fetch from instruction memory

RD - Instruction decode and register read

EX - Execute: ALU operation for data and address computation

MA - Data memory access - for write access, the register read at RD state is used

WB - Register write back

Consider the following sequence of instructions.

I_1: $L\ R_0$, loc_1; $R_0 <= M[\text{loc}_1]$
I_2: $A\ R_0$, R_0; $R_0 <= R_0 + R_0$
I_3: $S\ R_2$, R_0; $R_2 <= R_2 - R_0$

Let each stage take one clock cycle.

What is the number of clock cycles taken to complete the above sequence of instructions from the fetch of I_1?

(A) 8 (B) 10
(C) 12 (D) 15

10. Consider a non-pipelined processor with a clock rate of 2.5 gigahertz and average cycles per instruction of four. The same processor is upgraded to a pipelined processor with five stages; but due to the internal pipeline delay, the clock speed is reduced to 2 gigahertz. Assume that there are no stalls in the pipeline. The speed up achieved in this pipelined processor is _____. **[2015]**

11. Consider the sequence of machine instructions given below:

MUL	R_5, R_0, R_1
DIV	R_6, R_2, R_3
ADD	R_7, R_5, R_6
SUB	R_8, R_7, R_4

In the above sequence, R_0 to R_8 are general purpose registers. In the instructions shown, the first register stores the result of the operation performed on the second and the third registers. This sequence of instructions is to be executed in a pipelined instruction processor with the following 4 stages: (1) Instruction Fetch and Decode (IF), (2) Operand Fetch (OF), (3) Perform Operation (PO) and (4) Write back the result (WB). The IF, OF and WB stages take 1 clock cycle each for any instruction. The PO stage takes 1 clock cycle for ADD or SUB instruction, 3 clock cycles for MUL instruction and 5 clock cycles for DIV instruction. The pipelined processor uses operand forwarding from the PO stage to the OF stage. The number of clock cycles taken for the execution of the above sequence of instructions is _____. **[2015]**

12. Consider the following reservation table for a pipeline having the stages S_1, S_2 and S_3.

Time →

	1	2	3	4	5
S_1	X				X
S_2		X		X	
S_3			X		

The minimum average latency (MAL) is _____

[2015]

13. Consider the following code sequence having five instructions I_1 to I_5. Each of these instructions has the following format. **[2015]**

OP Ri, *Rj*, *Rk*

Where operation *OP* is performed on contents of registers *Rj* and *Rk* and the result is stored in register *Ri*.

I_1: ADD R_1, R_2, R_3
I_2: MUL R_7, R_1, R_3
I_3: SUB R_4, R_1, R_5
I_4: ADD R_3, R_2, R_4
I_5: MUL R_7, R_8, R_9

Consider the following three statements.

S_1: There is an anti-dependence instructions between instructions I_2 and I_5

S_2: There is an anti-dependence between Instructions I_2 and I_4

S_3: |Within an instruction pipeline an anti-dependence always creates one or more stalls

Which one of the above statements is/are correct?

(A) Only S_1 is true
(B) Only S_2 is true
(C) Only S_1 and S_3 are true
(D) Only S_2 and S_3 are true

14. The stage delays in a 4 - stage pipeline are 800, 500, 400 and 300 picoseconds. The first stage (with delay 800 picoseconds) is replaced with a functionally equivalent design involving two stages with respective delays 600 and 350 picoseconds. The throughput increase of the pipeline is ___ percent. **[2016]**

15. Consider a 3 GHz (gigahertz) processor with a three - stage pipeline and stage latencies τ_1,τ_2 and τ_3 such that $\tau_1 = 3\tau_2/4 = 2\tau_3$. If the longest pipelines stage is split into two pipeline stages of equal latency, the new frequency is ____ GHz, ignoring delays in the pipeline registers. **[2016]**

16. Instruction execution in a processor is divided into 5 stages, *Instruction Fetch* (IF), *Instruction Decode* (ID), *Operand Fetch* (OF), *Execute* (EX), and *Write Back* (WB). These stages take **5, 4, 20, 10**, and **3 nanoseconds (ns)** respective. A pipelined implement action of the processor requires buffering between each pair of consecutive stages with a delay of **2 ns**. Two pipelined implementations of the processor are contemplated:

(i) A navie pipeline implementation (NP) with 5 stages and
(ii) An efficient pipeline (EP) where the OF stage is divided into stages OF1 and OF2 with execution times of **12 ns** and **8 ns** respectively.

The speedup (correct to two decimal places) achieved by EP over NP in executing 20 independent instructions with no hazards is _____. **[2017]**

17. The instruction pipeline of a RISC processor has the following stages: Instruction Fetch (IF), Instruction Decode (ID), Operand Fetch (OF), Perform Operation (PO) and Write back (WB). The IF, ID, OF and WB stages take 1 clock cycle each for every instruction. Consider a sequence of 100 instructions. In the PO stage, 40 instructions take 3 clock cycles each, 35 instructions take 2 clock cycles each, and the remaining 25 instructions take 1 clock cycle each. Assume that there are no data hazards and no control hazards.

The number of clock cycles required for completion of execution of the sequence of instructions is _____. **[2018]**

Answer Keys

Exercises

Practice Problems 1

1. A	**2.** D	**3.** B	**4.** B	**5.** B	**6.** D	**7.** C	**8.** D	**9.** C	**10.** D
11. A	**12.** A	**13.** A	**14.** B	**15.** D	**16.** D	**17.** B	**18.** A	**19.** B	**20.** B
21. D									

Practice Problems 2

1. C	**2.** C	**3.** B	**4.** D	**5.** D	**6.** D	**7.** D	**8.** A	**9.** C	**10.** B
11. C	**12.** B	**13.** B	**14.** B	**15.** B	**16.** C	**17.** C	**18.** B	**19.** A	**20.** B

Previous Years' Questions

1. 4	**2.** C	**3.** 1.54	**4.** B	**5.** C	**6.** B	**7.** A	**8.** D	**9.** A	**10.** 3.2
11. 13	**12.** 3	**13** B	**14.** 33.0 : 34.0		**15.** 4	**16.** 1.51	**17.** 219		

Chapter 5

Cache and Main Memory, Secondary Storage

LEARNING OBJECTIVES

- *Characteristics of memory system*
- *Memory hierarchy*
- *Locality of reference*
- *Cache memory*
- *Basic operation of cache*
- *Elements of cache design*
- *Replacement algorithm*
- *Secondary storage*
- *Disk*
- *Diskette*
- *Magnetic tape*
- *Optimal memory*

Characteristics of Memory System

1. **Location:** The term refers to whether memory is internal or external to the computer. The location of memory may be
   - Processor
   - Internal (main)
   - External (secondary)
2. **Capacity:** The capacity of internal memory is expressed in terms of bytes. The capacity specified using
   - Word size
   - Number of words
3. **Unit of transfer**
   - For internal memory, the unit of transfer is equal to the number of data lines into and out of the memory module. The unit of transfer need not equal a word or an addressable unit.
   - For external memory, data are often transferred in much larger units than a word, and these are referred to as blocks.
4. **Access method:** The various methods of accessing units of data are

   (i) Sequential access: Memory is organized into units of data, called records.

   Example: Magnetic tapes

   (ii) Direct access: Individual blocks or records have a unique address based on physical location.
   Example: Magnetic disks

   (iii) Random access: Each addressable location in memory has a unique, physically wired-in addressing mechanism. The time to access a given location is independent of the sequence of prior accesses and is constant.
   Example: Main memory

   (iv) Associative: This is a random access type of memory that enables one to make a comparison of desired bit locations within a word for a specified match.

5. **Performance:** Three performance parameters are:

   (i) Access time (latency):
   - For random access memory, this is the time it takes to perform a read or write operation.
   - For non-random-access memory, access time is the time it takes to position the read-write mechanism at the desired location.

   (ii) Memory cycle time: For a random access memory it consists of the access time plus any additional time required before a second access can commence.

   (iii) Transfer rate: This is rate at which data can be transferred into or out of memory unit.

For Random access memory,

$$\text{Transfer rate} = \frac{1}{\text{Cycle Time}}$$

For non-random access memory, $T_N = T_A + \frac{N}{R}$

Where, T_N = Average time to read or write N-bits.
T_A = Average access time
N = Number of bits
R = Transfer rate in bits per second

6. **Physical type:** The physical type of a memory will be
   i. Semiconductor
   ii. Magnetic
   iii. Optical
   iv. Magneto-optimal
7. **Physical characteristics:** The memory may be
   - Volatile/non-volatile
   - Erasable/non-erasable
8. **Organization:** There is a trade–off among the three key characteristic of memory.
   i. Cost
   ii. Capacity
   iii. Access time

Memory Hierarchy

Consider the following memory hierarchy, which shows the various memory components.

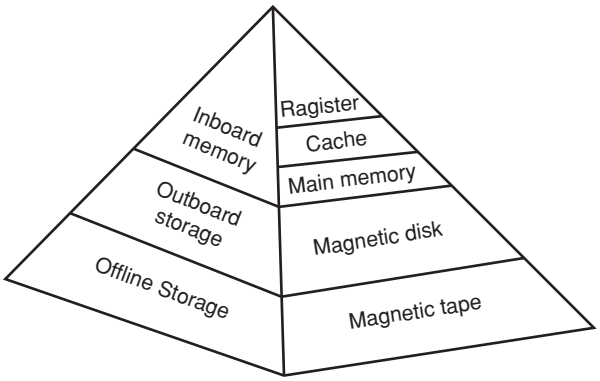


Figure 1 Memory Hierarchy

As one goes down the hierarchy, the following occur:

1. Decreasing cost per bit
2. Increasing capacity
3. Increasing access time
4. Decreasing frequency of access of the memory by the processor.

Locality of Reference

During the course of execution of a program, memory references by the processor, for both instructions and data, tend to cluster. This is referred as principal of locality.

(i) Registers: The fastest, smallest and most expensive type of memory consists of the registers internal to the processor.

(ii) Main memory: The principal internal memory system of the computer is main memory. Each location in main memory has a unique address.

(iii) Cache: Main memory is usually extended with a higher speed, smaller cache. The cache is not visible to the programmer or, indeed, to the processor. It is a device for staging the movement of data between main memory and processor registers to improve performance.

These three forms of memory are volatile and employ semi conductor technology.

(iv) Magnetic tapes and disks: Data are stored more permanently on external mass storage devices, of which the most common are hard disk and removable media.

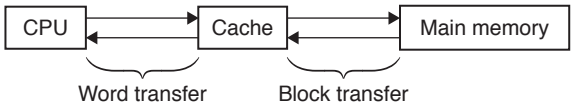
- External, non-volatile memory is also referred to as secondary or auxiliary memory.
- Used to store program and data files, which are visible to the programmer in the form of files and records.

Cache Memory

The locality of reference property states that over a short interval of time, the address generated by a typical program refer to a few localized areas of memory repeatedly, while the remainder of memory is accessed relatively infrequently (Because of frequent loops and subroutine calls).

If the active portions of the program and data are placed in a fast small memory, the average memory access time can be reduced, thus reducing the total execution time of the program. Such a fast small memory is referred to as a cache memory.

Cache memory is intended to give memory speed approaching that of the fastest memories available and at the same time provide a large memory size at the price of less expensive types of semiconductor memories. The following figure shows the structure of cache/main memory system.



The fundamental idea of cache organization is that by keeping the most frequently accessed instructions and data in the fast memory, the average memory access time will approach the access time of cache.

Basic Operation of Cache

- When the CPU need to access memory, the cache is examined. If the word is found in cache, it is read otherwise main memory is accessed to read the word.

- The performance of cache memory is measured in terms of hit ratio.
- When the CPU refers to memory and find the word in cache, it is called hit.
- If the word is not found in cache and is in Main Memory, it is called miss.

$$\text{Hit ratio} = \frac{\text{hits}}{\text{hits} + \text{misses}}$$

Average access time = $hc + (1 - h)(c + m)$m

Where, $c \rightarrow$ Cache access time

$m \rightarrow$ Main memory access time

$h \rightarrow$ hit ratio

- Let main memory consists of up to 2^n addressable words, with each word having a unique n-bit address.
- For mapping purposes, this memory is considered to consist of a number of fixed length blocks of K words each.

$$\therefore \text{ Number of blocks } (M) = \frac{2^n}{K}$$

- The cache consists of C lines.
- Each line contains K words, plus a tag of a few bits.
- The number of words in a line is referred to as the line size.
- The number of lines is considerably less than the number of main memory blocks i.e., $C << M$.
- Each line includes a tag that identifies which particular block is currently being stored.

  The tag is usually a portion of the main memory address.

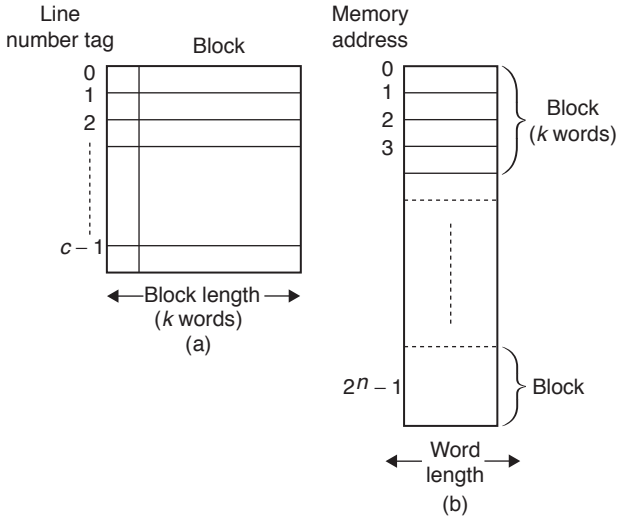


Figure 2 (a) Cache, (b) Main memory

Elements of Cache Design

1. Cache size
2. Mapping function
   - Direct
   - Associative
   - Set-associative
3. Replacement algorithm
4. Write policy
   - Write through
   - Write back
   - Write once
5. Line size
6. Number of caches
   - Single or two level
   - Unified or split

Cache size

The size of the cache to be small enough so that the overall average cost per bit is close to that of main memory alone and large enough so that the overall average access time is close to that of the cache alone.

Mapping function

Because there are fewer cache lines than main memory blocks, an algorithm is needed for mapping main memory blocks into cache lines. Three techniques can be used for mapping.

(i) Direct
(ii) Associative
(iii) Set-associative

Direct mapping Maps each block of main memory into only one possible cache line. Figure 2 illustrates the general mechanism. The mapping is expressed as

$i = j$ modulo m, where
i = cache line number
j = main memory block number
m = number of lines in the cache

For purpose of cache access, each main memory address can be viewed as consisting of three fields.

- The least significand w bits identify a unique word or byte within a block of main memory.
- The remaining s-bits specify one of the 2^s blocks of main memory. The cache logic interpret these s-bits as a tag of s-r bits. (most significand portion)
- A line field of r-bits, to identify one of 2^r lines.

To summarize,

Address length = $(s + w)$ bits

Number of Addressable units = 2^{s+w} words or bytes

Block size = line size = 2^w words or bytes

Number of blocks in main memory = $\frac{2^{s+w}}{2^w} = 2^s$ Number of lines in cache = $M = 2^r$.

Size of Tag = $(s - r)$ bits

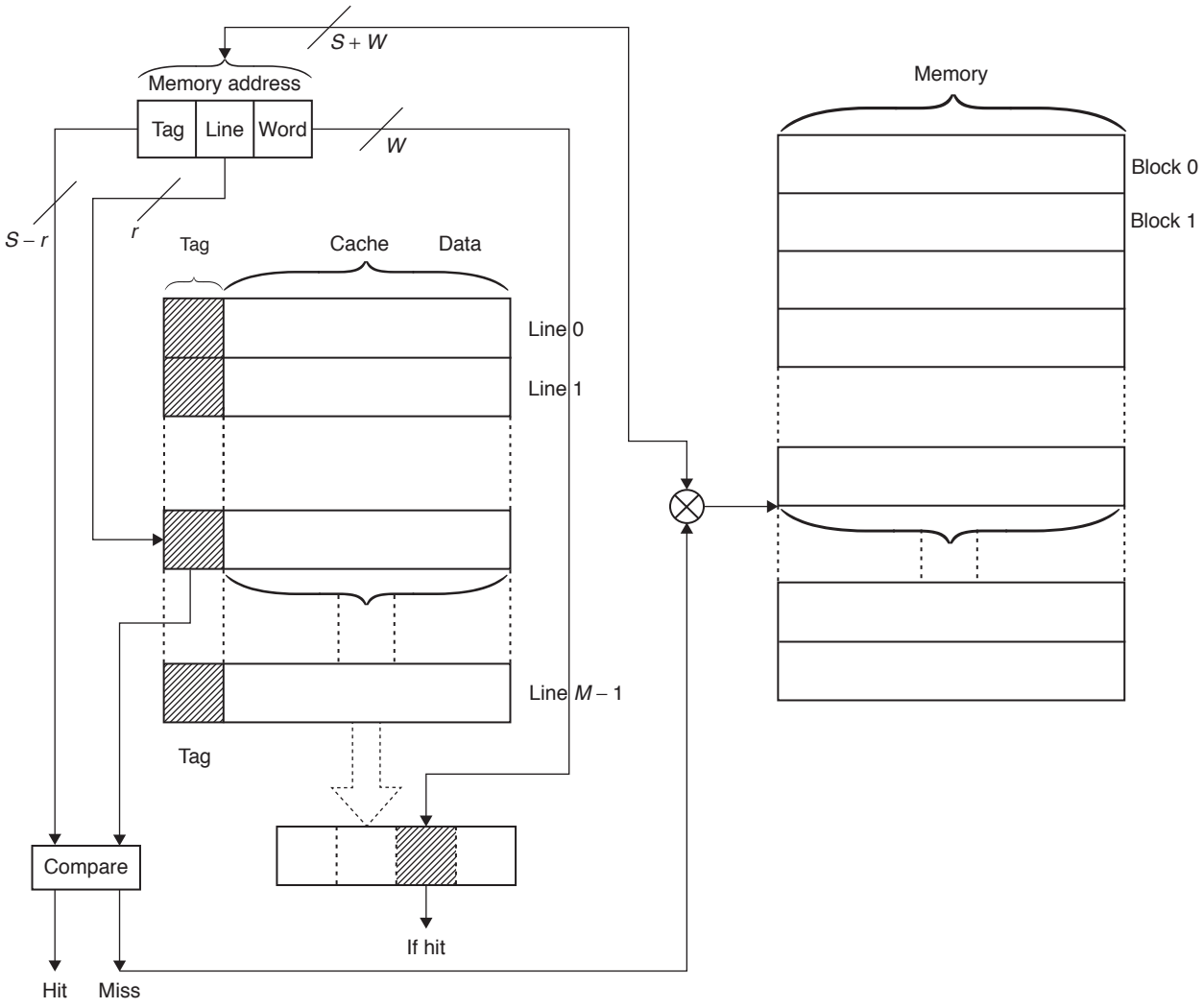


Figure 3 Direct Mapping

The effect of this mapping is that blocks of main memory are assigned to lines of the cache as follows:

Cache Line	Main Memory Blocks Assigned
0	0, m, $2m$, … $2^s - m$
1	1, $m+1$, $2m+1$, … $2^s - m + 1$
.	
.	
.	
.	
$m-1$	$m-1$, $2m-1$, $3m-1$,… $2^s - 1$

Example 1: Let cache capacity = 64 KB
Line size = 4 B
Main memory capacity = 16 MB = 2^{24} B
Using direct mapping, Address length = $s + w$ = 24-bits
Block size = 2^2 B

$$\text{Number of blocks in main memory} = \frac{2^{24}}{2^2} = 2^{22}$$

$$\text{Number of lines in cache} = m = 2^r = \frac{2^{16}}{2^2} = 2^{14}$$

∴ Size of tag = $s - r = 22 - 14 = 8$
∴ Main memory address =

Tag	Line	Word
8	14	2

The mapping becomes

Cache Line	Starting Memory Address of Blocks (Hexa)
0	00000, 010000, … FF0000
1	000004, 010004, … FF0004
.	.
.	.
.	.
.	.
$2^{14} - 1$	00FFFC, 01FFFC,… FFFFFC

Note: No two blocks that map into the same line number have the same tag number.

Advantages:
- Simple and cheap
- The tag field is short; only those bits have to be stored which are not used to address the cache.
- Access is very fast.

Disadvantages: A given block fits into a fixed cache location, i.e., a given cache line will be replaced whenever there is a reference to another memory block which fits to the same line, regardless what the status of the other cache line is.

This can produce a low hit ratio, even if only a very small part of the cache is effectively used.

Associative mapping This technique overcomes the disadvantage of direct mapping by permitting each main memory block to be loaded into any line of the cache. Here the cache control logic interprets a memory address as two fields.

1. Tag
2. Word

Figure shows associative mapping technique:

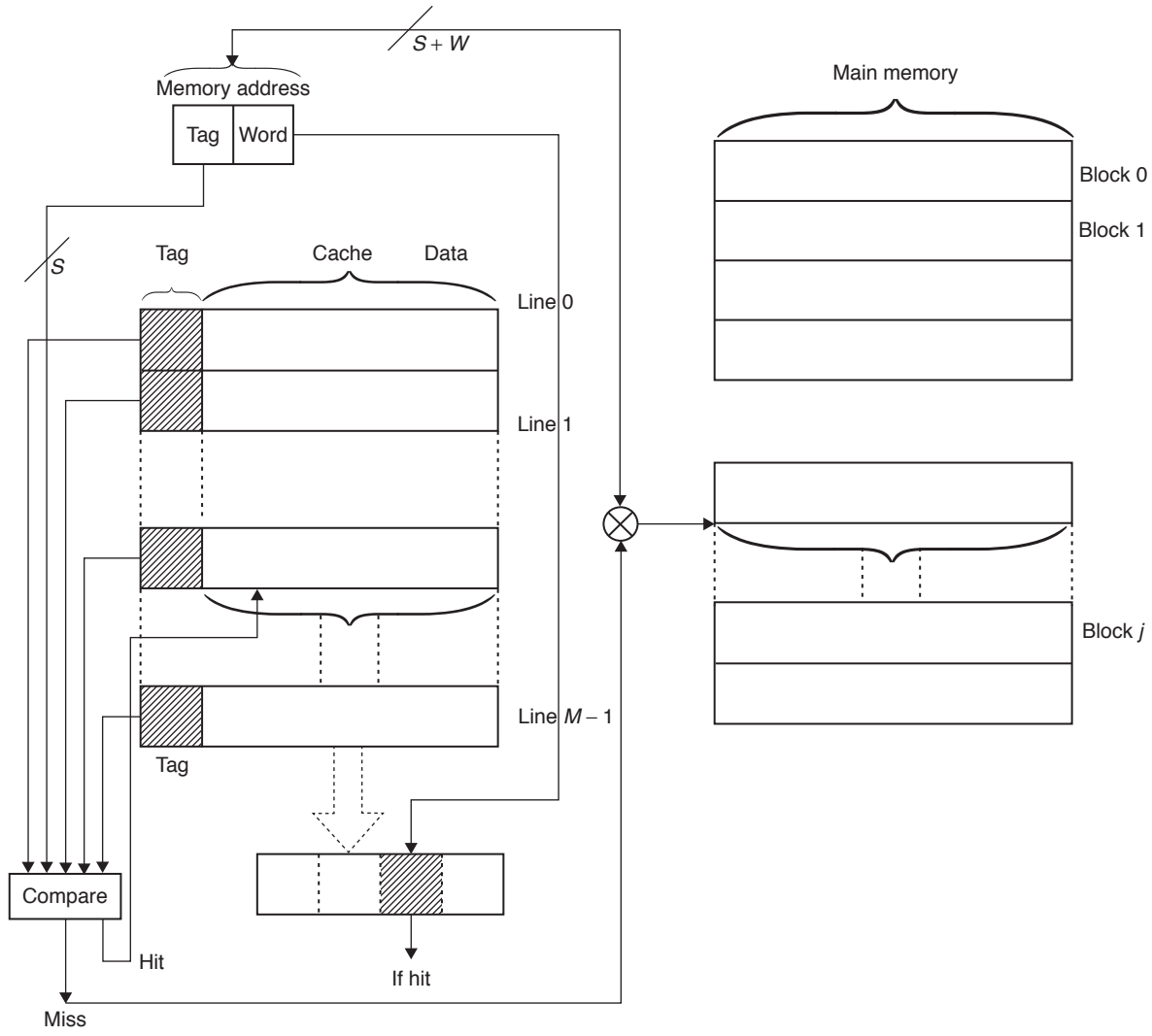


Figure 4 Associative mapping

To determine whether a block is in the cache, the cache control logic must simultaneously examine every line tag for a match. No field in the address corresponds to line number, so that the number of lines in the cache is not determined by the address format.

To summarize,

Address length = $(s + w)$ bits
Number of addressable units = 2^{s+w} words or bytes
Block size = line size = 2^w words or bytes
Number of blocks in main memory = $\frac{2^{s+w}}{2^w} = 2^S$
Number of lines in cache = undetermined
Size of tag = s-bits

Example 2: Cache size = 64 KB
Line size = 4 B
Main memory capacity = 16 MB
Number of blocks in main memory $= \frac{2^{24}}{2^2}$
$= 2^{22}$.
∴ Size of tag = 24 – 2 = 22-bits
For example, the tag of the hexadecimal main memory address 16339C is 058CE7
Main memory address =

Tag	Word
22	2

Advantages: Associative mapping provides the highest flexibility concerning the line to be replaced when a new block is read into a cache.

Disadvantages:
- Complex
- The tag field is long
- Fast access can be achieved only using high performance associative memories for the cache, which is difficult and expensive.

Set-associative mapping: It exhibits strengths of both the direct and associative approaches and reduces their disadvantages.
Here the cache is divided into V sets, each of which consists of K lines

i.e., $m = V \times K$
$i = j$ modulo V
Where i = cache set number
j = main memory block number
m = number of lines in cache

As there are K lines in each set, this is referred as K-way set associative mapping. The cache control logic interprets a memory address simply as three fields.

1. Tag
2. Set
3. Word

The d set bits specify one of $V = 2^d$ sets. The S-bits of the tag and the set fields specify one of the 2^S blocks of main memory.
Figure 3 shows Set-associative mapping.

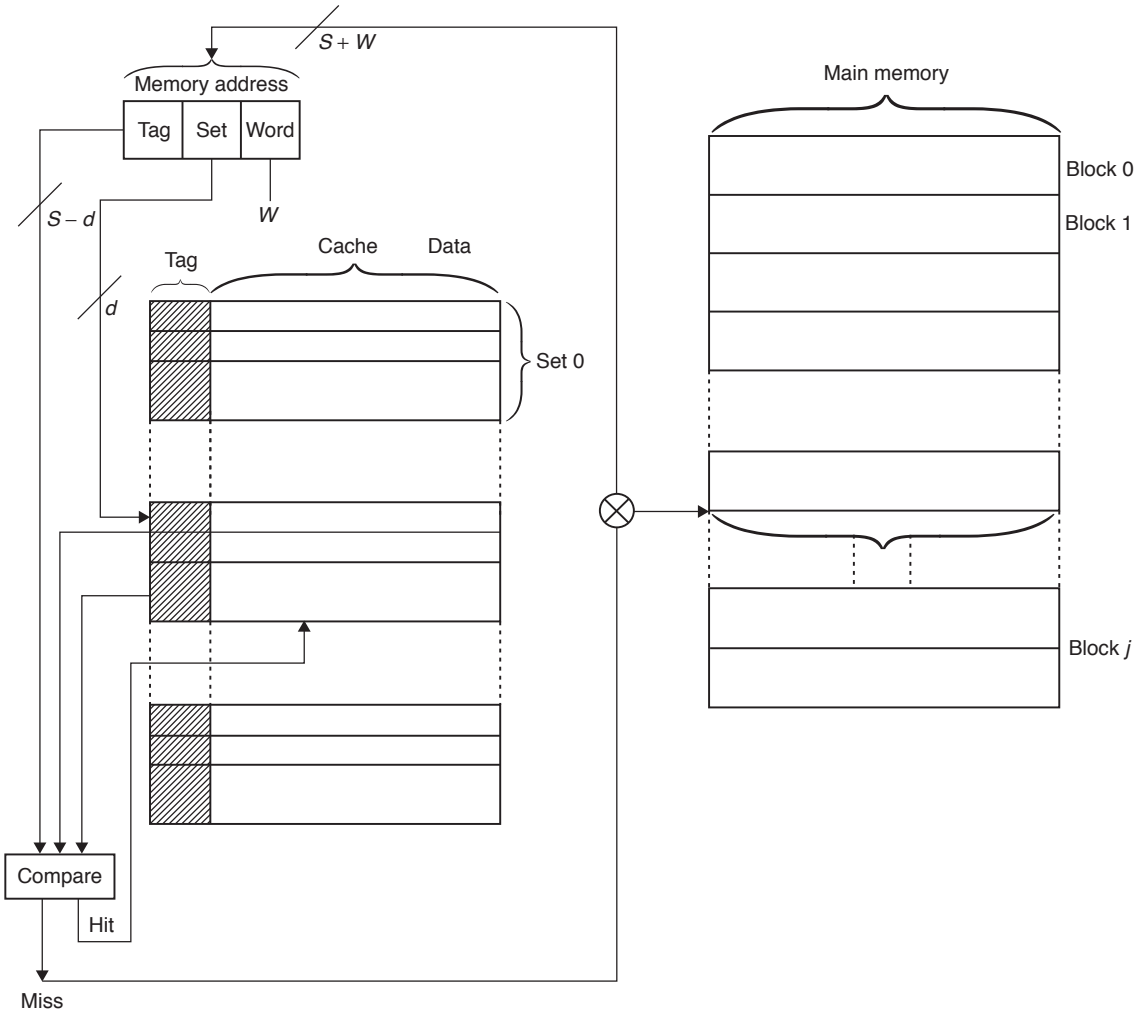


Figure 5 *K*-way set associative cache

Here the tag in a memory address is much smaller and is only compared to the K tags within a single set. To summarize,

Address length = $(s + w)$ bits

Number of Addressable units = 2^{s+w} words or bytes.

Block size = line size = 2^w words or bytes.

Number of blocks in main memory = $\frac{2^{s+w}}{2^w} = 2^s$

Number of lines in set = K

Number of sets $V = 2^d$

Number of lines in cache = $KV = K \times 2^d$

$\therefore$ Size of tag = $(s - d)$ bits

Example 3: Cache capacity = 64 KB
Block size = 4 B
Main memory capacity = 16 MB

Number of blocks in main memory $= \frac{2^{24}}{2^2} = 2^{22}$

For 2-way set associative mapping,
Number of lines in a set $K = 2$
Number of sets = $V = 2^d$

Number of lines in cache = $K \times 2^d = \frac{2^{16}}{2^2}$

$= 2^{14}$

$\Rightarrow \quad 2 \times 2^d = 2^{14}$

$\Rightarrow \quad 2^d = 2^{13}$

$\Rightarrow \quad d = 13$

$\therefore$ Size of Tag = 22 − 13 = 9

Hence main memory address =

Tag	Set	Word
9	13	2

In practice, 2 and 4-way set associative mapping are used with very good results. Larger sets do not produce further significant performance improvement.

If a set consist of a single line, i.e., it is direct mapping; If there is one single set consisting of all lines i.e., it is associative mapping.

Replacement algorithms

Once the cache has been filled, when a new block is brought into the cache, one of the existing blocks must be replaced. For direct mapping, there is only possible line for any particular block, and no choice is possible.

For associative and set associative techniques, a replacement algorithm is needed. Four of the most common replacement algorithms are

(i) LRU (Least recently used): Replaces the block in the set that has been in the cache longest with no reference to it.

(ii) FIFO (First-in-first-out): Replace the block in the set that has been in the cache longest.

(iii) LFU (Least frequently used): Replace the block in the set that has experienced the fewest references.

(iv) Random

Write policy

When a block that is resident in the cache is to be replaced, there are two cases to consider.

(i) If the old block in the cache has not been altered, then it may be over-written with a new block without first writing out the old block.

(ii) If at least one write operation has been performed on a word in that line of the cache, then main memory must be updated by writing the line of cache out of the block of memory before bringing in the new block.

The write policies are

(a) Write through: All write operations are made to main memory as well as to the cache, ensuring that main memory is always valid.
Drawback: Creates substantial memory traffic

(b) Write back: This technique minimizes memory writes. It updates are made only in the cache. When a block is replaced it is written back to main memory if and only if it is updated.
Drawback: There some portions of main memory are invalid and hence accesses by I/O modules can be allowed only through the cache.

Line size

Larger blocks reduce the number of blocks that fit into a cache. Because each block fetch overwrites older cache contents, a small number of blocks results in data being over written shortly after they are fetched.

As a block becomes larger, each additional word is farther from the requested word, therefore less likely to be needed in the near future.

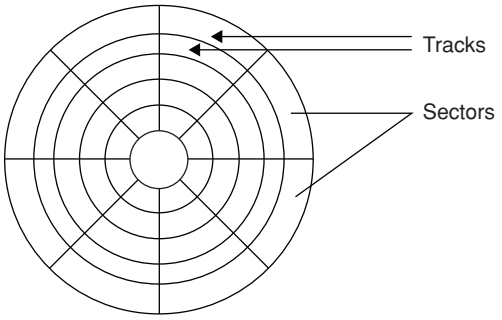
Number of caches

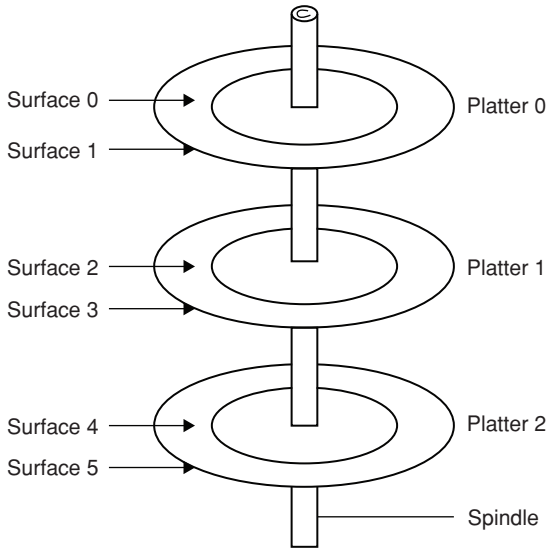
Multilevel caches We may have on-chip cache as well as external cache. This is a two level cache organization, with the internal cache designated as level 1, and external cache designated as level 2.

Secondary Storage

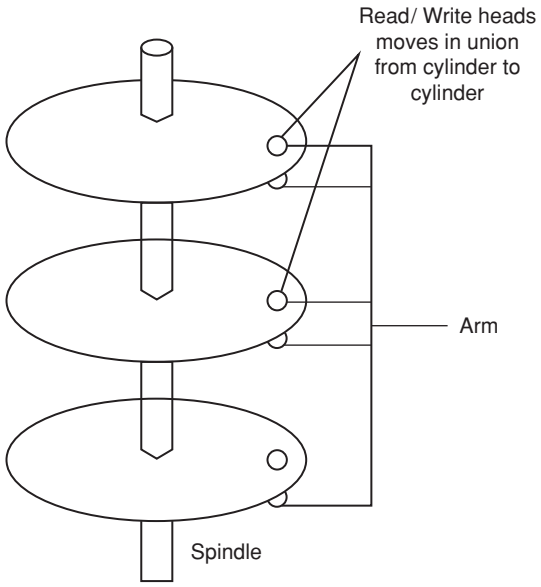
Disk

Disk consists of platters, each with two surfaces. Each surface consists of concentric rings called tracks. Each track consists of sectors separated by gaps.





Disk operation: The disk surface spins at a fixed rotational rate. There is a read/write head attached to the end of the arm and flies over the disk surface on a thin cushion of air. By moving radially the arm can position the read/write head over any track.



Disk access time: Average time to access some target sector:

$$T_{ae} = T_{\text{avg seek}} + T_{\text{avg rotation}} + T_{\text{avg transfer}}$$

Where $T_{\text{avg seek}}$ is typical 9 ms.

$$T_{\text{avg rotation}} = \frac{1}{2} \times \frac{1}{\text{RPMS}} \times 60 \text{ Sec/1min}$$

$$T_{\text{avg rotation}} = \frac{1}{\text{RPM}} \times \frac{1}{(\text{avg sector/track})} \times 60 \text{ Sec/1min}$$

Notes:

1. Seek time is the Time to position heads over cylinder containing target sectors ($T_{\text{avg seek}}$).
2. Rotational Latency is the time waiting for first bit of target sector to pass under read/write head. ($T_{\text{avg rotation}}$).
3. Transfer Time is the time to read the bits in the target sector ($T_{\text{avg transfer}}$).
   - Data are recorded on the surface of a hard disk made of metal coated with magnetic material.
   - The disks and the drive are usually built together and encased in an air tight container to protect the disk from pollutants such as smoke particle and dust. Several disks are usually started on a common drive shaft with each disk having its own read/write head.

Diskette

Data are recorded on the surface of a floppy disk made of polyester coated with magnetic material.

A special diskette drive must be used to access data stored in the floppy disk. It works much like a record turntable of Gramophone.

Main features

- Direct access
- Cheap
- Portable, convenient to use

Main Standards

- $5^1/_4$ inch capacity ≅ 360 KB/ disk
- $3^1/_2$ inch capacity ≅ 1.44 MB/disk (about 700 pages of A_4 text)

Magnetic Tape

Magnetic tape is made up from a layer of plastic which is coated with iron oxide. The oxide can be magnetized in different directions to represent data. The operation uses a similar principle as in the case of a tape recorder.

Main features

- Sequential access (access time about 1.55)
- High value of storage (50 MB/tape)
- Inexpensive

It is often used for Batch up or archive purpose.

Optimal Memory

CD-ROM (Compact disk ROM): The disk surface is imprinted with microscopic holes which record digital information. When a low-powered power beam shines on the surface, the intensity of the reflected light changes as it encounters a hole. The change is detected by a photo sensor and converted into a digital signal.

- Huge capacity: 775 MB/disk (≈550 diskette)
- Inexpensive replication, cheap production.
- Removable, read only.
- Long access time (could be half a second)

WORM (Write Once Read Memory) CD: A lower beam of modest intensity equipped in the disk drive is used to imprint the hole pattern.

- Good for archival storage by providing a permanent record of large volumes of data.

Erasable Optical Disk: Combination of Laser technology and magnetic surface technique.

- Can be repeatedly written and overwritten.
- High reliability and longer life than magnetic disks.

EXERCISE

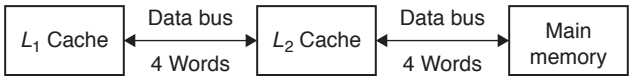
Practice Problems I

Directions for questions 1 to 20: Select the correct alternative from the given choices.

1. Find the number of bits in the cache index and tag for a direct mapped cache of size 32 KB with block size of 32 bytes. The CPU generates 48-bit addresses.
(A) 33,15 (B) 15,10
(C) 10,33 (D) 15,33

2. Given the cache access time is 200 ns and the memory access time is 400 ns. If the effective access time is 20% greater than the cache access time, what is the hit ratio?
(A) 80% (B) 20%
(C) 40% (D) 100%

3. A computer system has an L_1 cache, an L_2 cache and a main memory unit connected as shown below. The block size in L_1 cache is 2 words. The block size in L_2 cache is 8 words. The memory access times are 2 nanoseconds, 20 nanoseconds and 200 nanoseconds for L_1 cache, L_2 cache and main memory unit, respectively.



When there is a miss in L_1 cache and a hit in L_2 cache, a block is transferred from L_2 cache to L_1 cache. What is the time taken for this transfer?
(A) 22 ns (B) 44 ns
(C) 66 ns (D) 88 ns

4. In direct memory management, CPU references address of 15-bits. Main memory size is 512 * 8 and cache memory size is 128 * 8. Tag and line are respectively
(A) 9, 7 (B) 7, 9
(C) 15, 7 (D) 7, 15

5. Consider a cache with 64 blocks and a block size of 16 bytes. The byte address of 1200 maps to ____ block number.
(A) 10 (B) 11
(C) 64 (D) 16

6. In a cache memory, cache line is 64 bytes. The main memory has latency of 32 ns and bandwidth of 1 GB/sec. Then the time required to fetch the entire cache line from main memory is
(A) 32 ns (B) 64 ns
(C) 96 ns (D) 128 ns

7. A set associative cache consists of 64 lines or slots divided into four-line sets. Main memory contains 4K blocks of 128 word each. Then the number of bits present in tag, set and word fields are respectively.
(A) 7, 6, 7 (B) 6, 7, 7
(C) 4, 8, 7 (D) 8, 4, 7

8. A 2-way set-associative cache has lines of 32 bytes and a total size of 16 KB. The 32 MB main memory is byte addressable. Then which of the following two memory addresses mapped to same set?
(A) 10D6A32, 035C3A2
(B) 2A36D01, 2A3C530
(C) 10D63A2, 035C3A0
(D) 2A36D08, 0A3C538

9. Let the cache memory capacity is 64 KB and main memory capacity is 16 MB. Let block size is 4 bytes. Then the tag, line, word fields in hexadecimal notation for the main memory address cccccc using direct mapped cache will be
(A) cc, ccc, c (B) cc, 3333, 0
(C) cc, ccc, 0 (D) cc, 333, 30

10. Consider a 32-bit microprocessor that has an on-chip 16 KB four-way set associative cache. Assume that the cache has a line size of four 32-bit words. Then the word in the memory location ABCDE8F8 will be mapped to
(A) 143rd set (B) 815th set
(C) 255th set (D) 0th set

11. Given the following specifications for an external cache memory:
Four-way set associative, Line size of two 16-bit words; Able to accommodate a total of 4K 32-bit words from

main memory. Used with a 16-bit processor that issues 24-bit address. Then the number of bits used to represent set field is
(A) 2-bits (B) 10-bits
(C) 12-bits (D) 14-bits

12. Consider a machine with a byte addressable main memory of 2^{16} bytes and block size of 8 bytes. Assume that a direct mapped cache consisting of 32 lines is used with this machine. Then in what line would bytes with the address 1100 0011 0011 0100 is stored?
(A) slot 3 (B) slot 4
(C) slot 6 (D) slot 12

13. A computer system contains a main memory of 32 K 16-bit words. It also has a 4 K-word cache divided into four line sets with 64 words per line. The processor fetches words from locations 0, 1, 2…, 4351 in that order. It then repeats this fetch sequence 10 more times. The cache is 10 times faster than main memory. Then the improvement resulting from the use of the cache is (assume an LRU policy is used for block replacement)
(A) 0.63 (B) 0.45
(C) 1.21 (D) 2.18

14. Consider an L_1 cache with an access time of 1ns and a hit ratio of $H = 0.95$. Suppose that we can change the cache design such that we increase H to 0.98, but increase access time to 1.5ns. Which of the following condition is met for this change to result in improved performance?
(A) Next level memory access time must be less than 16.67
(B) Next level memory access time must be greater than 16.67
(C) Next level memory access time must be less than 50
(D) Next level memory access time must be greater than 50

15. Consider a single-level cache with an access time of 2.5 ns and a line size of 64 bytes and a hit ratio of $H = 0.95$. Main memory uses a block transfer capability that has a first-word (4 bytes) access time of 50 ns and an access time of 5ns for each word thereafter. What is the access time when there is a cache miss?
(A) 130 ns (B) 149.4 ns
(C) 2.375 ns (D) 8.875 ns

16. The tag, block and word fields of main memory address using direct mapping technique for 2048 main memory blocks, 128 blocks of cache memory and block size of 16:
(A) 4, 7, 4 (B) 7, 4, 4
(C) 11, 7, 4 (D) Data insufficient

17. Let H_1 is level 1 cache hit ratio, H_2 is level 2 cache hit ratio, C_1 is the time required to access Level 1 cache, C_2 is the time required to access Level 2 cache and M is the time required to access Main memory. Then the average access time required by the processor is
(A) $H_1C_1 + (1 - H_1)\, H_2(C_2) + (1 - H_1)\, (1 - H_2)\, (M)$
(B) $H_1C_1 + (1 - H_1)\, H_2(C_1 + C_2) + (1 - H_1)\, (1 - H_2)\, (C_1 + C_2 + M)$
(C) $H_1C_1 + H_1H_2(C_1 + C_2) + H_1H_2(C_1 + C_2 + M)$
(D) $H_1C_1 + (1 - H_1)\, H_2(C_1.C_2) + (1 - H_1)\, (1 - H_2)\, (C_1.C_2.M)$

18. If $p = 2^m$ be the number of lines in cache and $b = 2^n$ be the size of each block, then total words that can be stored in cache memory is given by
(A) 2^{m+n} (B) 2^{m-n}
(C) $m + n$ (D) $p + b$

19. Cache memory enhances
(A) memory capacity
(B) memory access time
(C) secondary storage capacity
(D) secondary storage access time

20. Which of the following property allows the processor to execute a number of clustered locations?
(A) Spatial (B) Temporal
(C) Inclusion (D) Coherence

Practice Problems 2

Directions for questions 1 to 20: Select the correct alternative from the given choices.

1. If average access time of CPU is 20 ns, access time of main memory is 110 ns and the cache access time is 10 ns. What is the hit ratio?
(A) 100% (B) 90%
(C) 80% (D) 70%

2. A hard disk spins at 180 revolutions per minute. What is the average rotational latency?
(A) 0.16 sec (B) 0.32 sec
(C) 0.2 sec (D) 0.4 sec

3. A disk pack have 16 surfaces, with 128 tracks per surface and 256 sectors per track. 512 bytes of data are stored in a bit serial manner in a sector. The number of bits required to specify a particular sector in the disk is
(A) 4 (B) 7
(C) 11 (D) 19

4. A disk has 19456 cylinders, 16 heads and 63 sectors per track. The disk spins at 5400 rpm. Seek time between adjacent tracks is 2 ms. Assuming the read/write head is already positioned at track 0, how long does it take to read the entire disk?
(A) 48 min (B) 58 min
(C) 64 min (D) 72 min

5. A certain moving arm disk storage with one head has following specifications:
Number of tracks/recording surface = 200
Disk Rotation Speed = 2400 rpm
Track storage capacity = 62500-bits
The average latency time (assuming that head can move from one track to another only by traversing the entire track) is
(A) 0.125 sec (B) 1.25 sec
(C) 0.0125 sec (D) 12.5 sec

6. In Memory management system, cache memory access time is 100 ns and main memory access time is 200 ns. Number of CPU references is 100 and number of hits is 10. Average access time is
(A) 150 ns (B) 100 ns
(C) 190 ns (D) 280 ns

7. The seek time of disk is 40 m sec. It rotates at the rate of 40 rps. The capacity of each track is 400 words. The access time is
(A) 50 m sec (B) 53 m sec
(C) 60 m sec (D) 63 m sec

8. An Associated cache and one million word main memory are divided into 256 word blocks. How many blocks are there?
(A) 2^8 (B) 2^{12}
(C) 2^{20} (D) 2^{28}

9. The average access time of a disk is
(A) Seek time + Rotational latency time
(B) Seek time
(C) Rotational latency + transfer time + seek time
(D) Rotation latency + transfer time.

10. What will be the size of the memory whose last memory location is FFFF?
(A) 64 k (B) 32 k
(C) 10 k (D) 24 k

11. Data from a cassette tape is obtained by ____ accessing method.
(A) Parallel (B) Serial
(C) Sequential (D) Random

12. For a memory system, the desirable characteristics is/are
(A) Speed and reliability
(B) Durability and compactness
(C) Low power consumption
(D) All of these

13. The memory that has the shortest access time is
(A) Magnetic bubble (B) Magnetic core memory
(C) Cache memory (D) RAM

14. Cache memory
(A) has greater capacity than RAM.
(B) enhances secondary storage access time.
(C) is faster to access than registers.
(D) is faster to access than main memory

15. Consider a disk pack with 16 surfaces 128 tracks per surface and 256 sectors per track. 512 bytes of data are stored in bit and serial manner, Then the capacity of the disk is
(A) 256 MB (B) 256 KB
(C) 512 MB (D) 64 MB

16. Principle of locality justifies the use of
(A) Cache (B) DMA
(C) Disk (D) RAM

17. The main memory of a computer has $2ab$ blocks while cache has $2a$ blocks. If the cache uses the set associative mapping scheme with two blocks per set, then block k of main memory maps to the set:
(A) (k mod b) of the cache (B) (k mod a) of cache
(C) (k mod $2a$) of cache (D) (k mod $2ab$) of cache

18. Which of the following factors do not affect the hit ration of cache?
(A) Block replacement algorithms.
(B) Block frame size
(C) Cycle counts
(D) Main memory size

19. In which of the following mapping function, there is no need of replacement algorithm?
(A) Direct Mapping
(B) Set-associative mapping
(C) Full associative mapping
(D) Both (A) and (B)

20. In a direct mapping, the index field equals to
(A) Sum of tag and word fields
(B) Sum of block and word fields
(C) Sum of tag an block fields
(D) Same as block field

Previous Years' Questions

1. Consider a small two-way set-associative cache memory, consisting of four blocks. For choosing the block to be replaced, use the least recently used (LRU) scheme. The number of cache misses for the following sequence of block addresses is 8, 12, 0, 12, 8 **[2004]**

(A) 2 (B) 3
(C) 4 (D) 5

2. Consider a direct mapped cache of size 32 KB with block size 32 bytes. The CPU generates 32 bit addresses. The number of bits needed for cache indexing and the number of tag bits are respectively. **[2005]**

(A) 10, 17 (B) 10, 22
(C) 15, 17 (D) 5, 17

Common data for questions 3 and 4: Consider two cache organizations: The first one is 32 KB 2-way set associative with 32-byte block size. The second one is of the same size but direct mapped. The size of an address is 32 bits in both cases. A 2-to-1 multiplexer has a latency of 0.6 ns while a k-bit comparator has a latency of k/10 ns. The hit latency of the set associative organization is h_1 while that of the direct mapped one is h_2.

3. The value of h_1 is: **[2006]**

(A) 2.4 ns (B) 2.3 ns
(C) 1.8 ns (D) 1.7 ns

4. The value of h_2 is: **[2006]**

Data for question 5: Consider a machine with a byte addressable main memory of 2^{16} bytes. Assume that a direct mapped data cache consisting of 32 lines of 64 bytes each is used in the system. A 50 × 50 two-dimensional array of bytes is stored in the main memory starting from memory location 1100 H. Assume that the data cache is initially empty. The complete array is accessed twice. Assume that the contents of the data cache do not change in between the two accesses.

5. Which of the following lines of the data cache will be replaced by new blocks in accessing the array for the second time? **[2007]**

(A) line 4 to line 11 (B) line 4 to line 12
(C) line 0 to line 7 (D) line 0 to line 8

6. For inclusion to hold between two cache levels L_1 and L_2 in a multi-level cache hierarchy, which of the following are necessary? **[2008]**

(i) L_1 must be a write-through cache
(ii) L_2 must be a write-through cache
(iii) The associativity of L_2 must be greater than that of L_1
(iv) The L_2 cache must be atleast as large as the L_1 cache

(A) (iv) only (B) (i) and (iv) only
(C) (i), (ii) and (iv) only (D) (i), (ii), (iii) and (iv)

Common data for questions 7, 8 and 9: Consider a machine with a 2-way set associative data cache of size 64K-bytes and block size 16-bytes. The cache is managed using 32-bit virtual addresses and the page size is 4Kbytes. A program to be run on this machine begins as follows:

```
double ARR [1024] [1024] ;
       int i, j;
/* Initialize array ARR to 0.0 */
 for (i = 0; i < 1024; i++)
 for (j = 0; j < 1024; j++)
 ARR [i] [j] = 0.0;
```

The size of double is 8 Bytes. Array ARR is located in memory starting at the beginning of virtual page 0XFF000 and stored in row major order. The cache is initially empty and no pre-fetching is done. The only data memory references made by the program are those to array ARR.

7. The total size of the tags in the cache directory is **[2008]**

(A) 32K-bits (B) 34K-bits
(C) 64K-bits (D) 68K-bits

8. Which of the following array elements has the same cache index as ARR [0] [0]? **[2008]**

(A) ARR [0] [4] (B) ARR [4] [0]
(C) ARR [0] [5] (D) ARR [5] [0]

9. The cache hit ratio for this initialization loop is **[2008]**

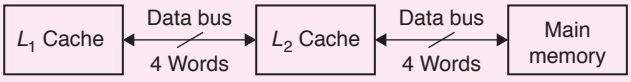
(A) 0% (B) 25%
(C) 50% (D) 75%

10. Consider a 4-way set associative cache (initially empty) with total 16 cache blocks. The main memory consists of 256 blocks and the request for memory blocks is in the following order: **[2009]**
0, 255, 1, 4, 3, 8, 133, 159, 216, 129, 63, 8, 48, 32, 73, 92, 155.

Which one of the following memory block will NOT be in cache if LRU replacement policy is used?

(A) 3 (B) 8
(C) 129 (D) 216

Common data questions 11 and 12: A computer system has an L_1 cache, an L_2 cache, and a main memory unit connected as shown below. The block size in L_1 cache is 4 words. The block size in L_2 cache is 16 words. The memory access times are 2 nanoseconds. 20 nanoseconds and 200 nanoseconds, for L_1 cache, L_2 cache and main memory unit respectively.



11. When there is a miss in L_1 cache and a hit in L_2 cache, a block is transferred from L_2 cache to L_1 cache. What is the time taken for this transfer? **[2010]**
(A) 2 ns (B) 20 ns
(C) 22 ns (D) 88 ns

12. When there is a miss in both L_1 cache and L_2 cache, first a block is transferred from main memory to L_2 cache, and then a block is transferred from L_2 cache to L_1 cache. What is the total time taken for these transfers? **[2010]**
(A) 222 ns (B) 888 ns
(C) 902 ns (D) 968 ns

13. An 8 KB direct-mapped write-back cache is organized as multiple blocks, each of size 32 bytes. The processor generates 32-bit addresses. The cache controller maintains the tag information for each cache block comprising of the following.
1 Valid bit
1 Modified bit
As many bits as the minimum needed to identify the memory block mapped in the cache.
What is the total size of memory needed at the cache controller to store meta-data (tags) for the cache? **[2011]**
(A) 4864 bits (B) 6144 bits
(C) 6656 bits (D) 5376 bits

Common data for questions 14 and 15: A computer has a 256 KB, 4-way set associative, write back data cache with block size of 32 bytes. The processor sends 32 bit addresses to the cache controller. Each cache tag directory entry contains, in addition to address tag, 2 valid bits, 1 modified bit and 1 replacement bit.

14. The number of bits in the tag field of an address is **[2012]**
(A) 11 (B) 14
(C) 16 (D) 27

15. The size of the cache tag directory is **[2012]**
(A) 160 K-bits (B) 136 K-bits
(C) 40 K-bits (D) 32 K-bits
(A) 2.4 ns (B) 2.3 ns
(C) 1.8 ns (D) 1.7 ns

16. In a k-way set associative cache, the cache is divided into v sets, each of which consists of k lines. The lines of a set are placed in sequence one after another. The lines in set s are sequenced before the lines in set $(s + 1)$. The main memory blocks are numbered 0 onwards. The main memory block numbered j must be mapped to any one of the cache lines from **[2013]**
(A) $(j \bmod v) * k$ to $(j \bmod v) * k + (k - 1)$
(B) $(j \bmod v)$ to $(j \bmod v) + (k - 1)$
(C) $(j \bmod k)$ to $(j \bmod k) + (v - 1)$
(D) $(j \bmod k) * v$ to $(j \bmod k) * v + (v - 1)$

17. An access sequence of cache block addresses is of length N and contains n unique block addresses. The number of unique block addresses between two consecutive accesses to the same block address is bounded above by K. what is the miss ratio if the access sequence is passed through a cache of associativity $A \geq K$ exercising least-recently used replacement policy? **[2014]**
(A) n/N (B) $1/N$
(C) $1/A$ (D) K/n

18. A 4-way set -associative cache memory unit with a capacity of 16 KB is built using a block size of 8 words. The word length is 32-bits. The size of the physical address space is 4 GB. The number of bits for the TAG field is _____. **[2014]**

19. In designing a computer's cache system, the cache block (or cache line) size is an important parameter. Which one of the following statements is correct in this context? **[2014]**
(A) A smaller block size implies better spatial locality.
(B) A smaller block size implies a smaller cache tag and hence lower cache tag overhead.
(C) A smaller block size implies a larger cache tag and hence lower cache hit time.
(D) A smaller block size incurs a lower cache miss penalty.

20. Consider a main memory system that consists of 8 memory modules attached to the system bus, which is one word wide. When a write request is made, the bus is occupied for 100 nanoseconds (ns) by the data, address, and control signals. During the same 100 ns, and for 500 ns thereafter, the addressed memory module executes one cycle accepting and storing the data. The (internal) operation of different memory modules may overlap in time, but only one request can be on the bus at any time. The maximum number of stores (of one word each) that can be initiated in 1 millisecond is _____. **[2014]**

21. If the associativity of processor cache is doubled while keeping the capacity and block size unchanged, which one of the following is guaranteed to be NOT affected? **[2014]**
(A) Width of tag comparator
(B) Width of set index decoder
(C) Width of way selection multiplexer
(D) Width of processor to main memory data bus

22. The memory access time is 1 nanosecond for a read operation with a hit in cache, 5 nanoseconds for a read operation with a miss in cache, 2 nanoseconds for a write operation with a hit in cache and 10 nanoseconds for a write operation with a miss in cache. Execution of a sequence of instructions involves 100 instruction fetch operations. 60 memory operand read operations

and 40 memory operand write operations. The cache - hit ratio is 0.9. The average memory access time (in nanoseconds) in executing the sequence of instructions is _____. **[2014]**

23. Assume that for a certain processor, a read request takes 50 nanoseconds on a cache miss and 5 nanoseconds on a cache hit. Suppose while running a program, it was observed that 80% of the processor's read requests result in a cache hit. The average read access time in nanoseconds is _____ **[2015]**

24. A computer system implements a 40-bit virtual address, page size of 8 kilobytes, and a 128-entry translation look-aside buffer (TLB) organized into 32 sets each having four ways. Assume that the TLB tag does not store any process id. The minimum length of the TLB tag in bits is _____ **[2015]**

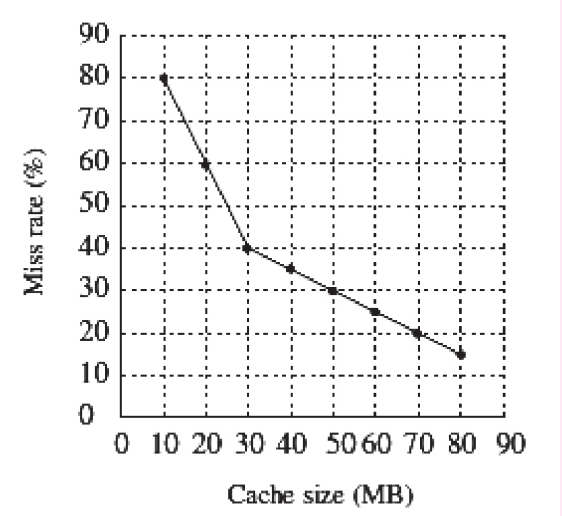
25. Consider a machine with a byte addressable main memory of 2^{20} bytes, block size of 16 bytes and a direct mapped cache having 2^{12} cache lines. Let the addresses of two consecutive bytes in main memory be $(E201F)_{16}$ and $(E2020)_{16}$. What are the tag and cache line address (in hex) for main memory address $(E201F)_{16}$? **[2015]**

(A) E, 201 (B) F, 201
(C) E, E20 (D) 2, 01F

26. The width of the physical address on a machine is 40 bits. The width of the tag field in a 512KB 8-way set associative cache is ____ bits. **[2016]**

27. A file system uses an in - memory cache to cache disk blocks. The miss rate of the cache is shown in the figure. The latency to read a block from the cache is 1 ms and to read a block from the disk is 10ms. Assume that the cost of checking whether a block exists in the cache is negligible. Available cache sizes are in multiples of 10MB.

The smallest cache size required to ensure an average read latency of less than 6 ms is ____ MB. **[2016]**



28. Consider a two-level cache hierarchy with L1 and L2 caches. An application incurs **1.4** memory accesses per instruction on average. For this application, the miss rate of L1 cache is **0.1**; the L2 cache experiences on average 7 misses per 1000 instructions. The miss rate of L2 expressed correct to two decimal places is _____. **[2017]**

29. Consider a 2-way set associative cache with 256 blocks and uses LRU replacement. Initially the cache is empty. Conflict misses are those misses which occur due to contention of multiple blocks for the same cache set. Compulsory misses occur due to first time access to the block. The following sequence of accesses to memory blocks

(0, 128, 256, 128, 0, 128, 256, 128, 1, 129, 257, 129, 1, 129, 257, 129)

is repeated 10 times. The number of *conflict misses* experienced by the cache is _____. **[2017]**

30. A cache memory unit with capacity of N words and block size of B words is to be designed. If it is designed as a direct mapped cache, the length of the TAG field is 10 bits. If the cache unit is now designed as a 16-way set-associative cache, the length of the TAG filed is _____bits. **[2017]**

31. In a two-level cache system, the access times of L_1 and L_2 caches are 1 and 8 clock cycles, respectively. The miss penalty from the L_2 cache to main memory is 18 clock cycles. The miss rate of L_1 cache is twice that of L_2. The average memory access time (AMAT) of this cache system is 2 cycles. The miss rates of L_1 and L_2 respectively are: **[2017]**

(A) 0.111 and 0.056 (B) 0.056 and 0.111
(C) 0.0892 and 0.1784 (D) 0.1784 and 0.0892

32. The read access times and the hit ratios for different caches in a memory hierarchy are as given below.

Cache	Read access time (in nanoseconds)	Hit ratio
I-cache	2	0.8
D-cache	2	0.9
L2-cache	8	0.9

The read access time of main memory is 90 nanoseconds. Assume that the caches use the referred- word-first read policy and the write back policy. Assume that all the caches are direct mapped caches. Assume that the dirty bit is always 0 for all the blocks in the caches. In execution of a program. 60% of memory reads are for instruction fetch and 40% are for memory operand fetch. The average read access time in nanoseconds (up to 2 decimal places) is _____. **[2017]**

33. Consider a machine with a byte addressable main memory of 2^{32} bytes divided into blocks of size 32

bytes. Assume that a direct mapped cache having 512 cache lines is used with this machine. The size of the tag field in bits is_____. **[2017]**

34. The size of the physical address space of a processor is 2^P bytes. The word length is 2^W bytes. The capacity of cache memory is 2^N bytes. The size of each cache block is 2^M words. For a K-way set-associative cache memory, the length (in number of bits) of the tag field is: **[2018]**

(A) $P - N - \log_2 K$
(B) $P - N + \log_2 K$
(C) $P - N - M - W - \log_2 K$
(D) $P - N - M - W + \log_2 K$

Answer Keys

Exercises

Practice Problems 1

1. D	**2.** A	**3.** D	**4.** A	**5.** B	**6.** C	**7.** D	**8.** C	**9.** B	**10.** A
11. B	**12.** C	**13.** D	**14.** B	**15.** A	**16.** A	**17.** B	**18.** A	**19.** B	**20.** A

Practice Problems 2

1. B	**2.** A	**3.** D	**4.** B	**5.** C	**6.** D	**7.** B	**8.** B	**9.** C	**10.** A
11. C	**12.** D	**13.** C	**14.** D	**15.** A	**16.** A	**17.** B	**18.** D	**19.** A	**20.** B

Previous Years' Questions

1. C	**2.** A	**3.** A	**4.** D	**5.** C	**6.** B	**7.** D	**8.** B	**9.** C	**10.** D
11. D	**12.** D	**13.** D	**14.** C	**15.** A	**16.** A	**17.** A	**18.** 20	**19.** D	
20. 10000		**21.** D	**22.** 1.68	**23.** 14	**24.** 22	**25.** A	**26.** 24	**27.** 30	**28.** 0.05
29. 76	**30.** 14	**31.** A	**32.** 4.72	**33.** 18	**34.** B				

Test

Computer Organization and Architecture

Time: 60 min.

Directions for questions 1 to 30: Select the correct alternative from the given choices.

1. Which of the following register keeps track of instruction execution sequence?
(A) Accumulator (B) Program counter
(C) Stack pointer (D) Instruction register

2. Consider the following Register Transfer Language:

$$R_1 \leftarrow R_1 + M[R_2 + R_3]$$

Where R_1, R_2 and R_3 are the CPU registers and 'M' is a memory location in primary memory, which addressing mode is suitable for above register transfer language?
(A) Indirect (B) Direct
(C) Indexed (D) Displacement

3. Which of the following is/are advantage(s) of using a multiple-bus architecture over a single-bus architecture?
(i) Multiple-bus architecture reduces propagation delay.
(ii) Multiple-bus architecture reduces bottleneck effects.
(A) (i) only (B) (ii) only
(C) Both (i) and (ii) (D) Neither (i) nor (ii)

4. Which of the following statement is false with respect to Booth's Multiplication Algorithm?
(i) Corrections required for the final result.
(ii) Sign bit is protected due to internal arithmetic shift.
(iii) More space required to maintain the sum.
(A) (i), (ii) only (B) (ii), (iii) only
(C) (i), (iii) only (D) (i), (ii), (iii)

5. After selective complement of $A = 1100$ with $B = 0101$, the resultant A will be
(A) 0000 (B) 1100
(C) 0101 (D) 1001

6. Which type of shift operation always keeps the sign bit unchanged?
(A) Logical shift (B) Arithmetic shift
(C) Circular shift (D) Any right shift

7. Consider the register transfer language instructions:

$AC \leftarrow M[R_1]$;

$R_1 \leftarrow R_1 + 1$;

Which addressing mode is specified by the instructions?
(A) Register addressing mode
(B) Register indirect mode
(C) Auto-increment mode
(D) Relative mode

8. Which of the following statement is true?
(A) Floating point representation is better than fixed point representation.
(B) Fixed point representation is better than floating point representation.
(C) Datapath is same as ALU.
(D) Both (A) and (C)

9. Which of the following statements correctly specifies about overflow?
(i) When adding two unsigned numbers the carry-out, from the MSB position serves as the overflow indicator.
(ii) Overflow can occur only by adding two signed numbers that have the same sign.
(A) (i) only (B) (ii) only
(C) Both (i) and (ii) (D) Neither (i) nor (ii)

10. A certain processor supports only the immediate and direct addressing modes. Which of the following programming language features cannot be implemented on this processor?
(A) Pointers (B) Arrays
(C) Records (D) All of these

11. The special purpose storage location(s) used by both ALU and CU are
(A) Decoders (B) Demultiplexers
(C) Registers (D) Buffers

12. Which of the following is a component of the datapath of Von Neumann machine?
(i) Registers
(ii) ALU input bus
(iii) ALU I/O registers
(A) (i), (ii) only (B) (ii), (iii) only
(C) (i), (ii), (iii) (D) None of these

13. Which of the following is/are false with respect to single-bus datapath?
(i) It is simplest and least expensive.
(ii) No limit on the amount of data transfer in a single clock cycle.
(A) (i) only (B) (ii) only
(C) Both (i) and (ii) (D) Neither (i) nor (ii)

14. If we have shifted the significant to the right by a single position, then
(A) Add one to the exponent
(B) Subtract one from the exponent
(C) Don't change the exponent
(D) Data insufficient

15. In which addressing mode, the effective address of the operand is generated by adding a constant value to the content of a register?
(A) Absolute mode (B) Indirect mode
(C) Immediate mode (D) Index mode

16. What is the number of instructions required to add 'n' numbers and store the result in memory using only one-address instructions?
(A) n (B) $n-1$
(C) $n+1$ (D) independent of n

17. Which unit of a computer system executes program, communicates with and often controls the operation of other subsystems?
(A) CPU (B) ALU
(C) I/O module (D) DMA

18. The multiplicand register and multiplier register of a hardware circuit implementing booth's algorithm have 1001 and 1100 respectively. The resultant will be
(A) 10011100 (B) 00011100
(C) 01101100 (D) 00010010

19. A floating point number has sign bit 0, Excess-64 exponent is 1010100 and fractional part is 0000000000011011. After converting this number to normalized form, the exponent (in decimal) will be
(A) 20 (B) 9
(C) 31 (D) 0

20. In IEEE floating point single precision representation, the number of bits in the fractional part is
(A) 24
(B) 23
(C) 32
(D) Depends on the architecture

21. After multiplying the binary numbers 010111 and 110110 using booth's multiplication algorithm, the resultant will be
(A) –1242 (B) 1242
(C) 230 (D) –230

22. The IEEE standard 754 single precision floating point representation of $(0.000001101101001 01)_2$ is.
(A) 0 10000111 11011010010100000000000
(B) 0 01111001 11011010010100000000000
(C) 0 10000110 10110100110100000000000
(D) 0 01111000 10110100101000000000000

23. How many clock cycles are required to perform two-operand operations using one bus datapath?
(A) 1 (B) 2
(C) 3 (D) Can't be determined

24. Which of the following is a rounding mode in IEEE754 standard?
(i) round to 0
(ii) round towards $+\infty$
(iii) round towards $-\infty$
(iv) round to nearest representable number
(A) (i), (iv) only (B) (ii), (iii) only
(C) (i), (ii) only (D) (i), (ii), (iii), (iv)

25. What is the normalized form of $0.00000110 \times 16^{101}$?
(A) 1.10×16^{107} (B) 1.10×16^{95}
(C) 0.110×16^{94} (D) 0.110×16^{106}

26. What is the biased representation of –7, using 4-bits for the bias?
(A) 0111 (B) 1111
(C) 0000 (D) 1001

27. What is the total resultant after adding $A = -7$ and $B = -6$ using signed two's complement representation?
(A) 0100 (B) 11101
(C) 1101 (D) Overflow occurs

28. What is the total number of additions and subtractions required using Booths multiplication algorithm for the multiplier 00011110?
(A) 1 (B) 2
(C) 30 (D) Can't be determined

Common data questions 29 and 30: Consider a 12-bit floating point format in which base $b = 2$, a 5-bit exponent e with a bias = 16 and 6-bit normalized mantissa m. Given two floating point numbers:

$A = 0\ 10001\ 011011$
$B = 1\ 01111\ 101010$

29. After adding A and B, the resultant will be
(A) 1 10001 000000
(B) 1 10001 000001
(C) 0 10001 000000
(D) 0 10001 000001

30. After subtracting B from A, the resultant will be
(A) 1 10001 110101 (B) 0 10001 110101
(C) 1 10001 110110 (D) 0 10001 110110

Answers Keys

1. B	**2.** C	**3.** C	**4.** C	**5.** D	**6.** B	**7.** C	**8.** A	**9.** C	**10.** D
11. C	**12.** C	**13.** B	**14.** A	**15.** D	**16.** C	**17.** A	**18.** B	**19.** B	**20.** B
21. D	**22.** D	**23.** B	**24.** D	**25.** B	**26.** C	**27.** D	**28.** B	**29.** C	**30.** D