Chapter 1

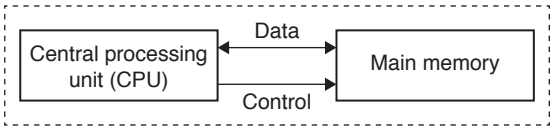
Machine Instructions, Addressing Modes

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.

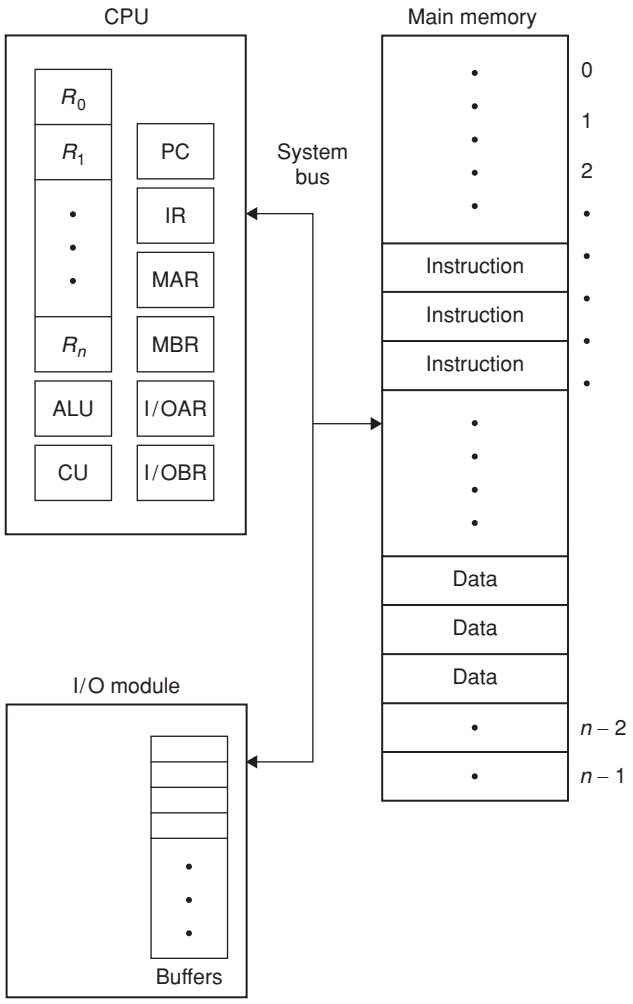
Example: Whether a computer will have a multiply instruction or not.

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.

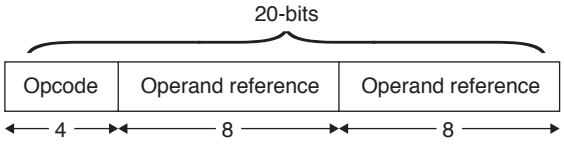
Elements of Machine Instructions

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.

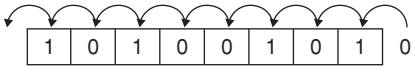
Example: AND, OR, NOT, EX-OR, Test, Compare, set control variables, shift, Rotate.
Let R_1 = 10100101, R_2 = 00001111 then
(R_1) AND (R_2) = 00000101.
AND is also called mask operation.
(R_1) OR (R_2) = 10101111.
NOT (R_1) = 01011010.
(R_1) EX-OR (R_2) = 10101010.

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

(a) Logical left shift:

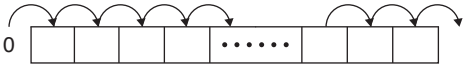
Here the bits of a word are shifted left. The left most bits is lost and 0 is shifted in right most bit position (i.e., bit empty).

Example: R_1 = 1010 0101
Logical left shift R_1:

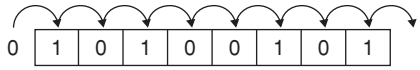


After left shift R_1 = 0100 1010.

(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.

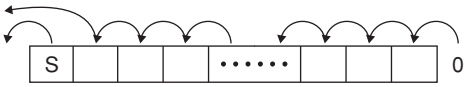


Example: R_1 = 1010 0101
Logical right shift R_1 = 0101 0010

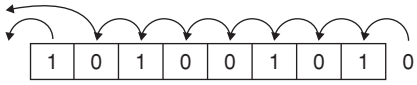


Logical shift operations are useful primarily for isolating fields within a word and also used to displace unwanted information.

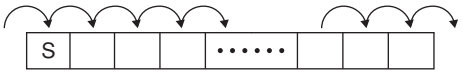
(c) Arithmetic left shift: Arithmetic shift operation treats the data as a signed integer and does not shift the sign bit. In Arithmetic left shift, a logical left shift is performed on all bits but the sign bit, which is retained.



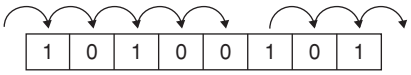
Example: $R_1 = 1010\ 0101$
Arithmetic Left shift $R_1 = 1100\ 1010$.



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



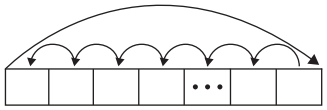
Example: $R_1 = 1010\ 0101$
Arithmetic Right shift $R_1 = 1101\ 0010$



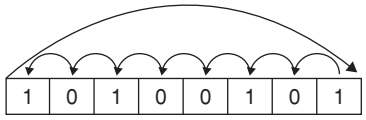
Notes:

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.

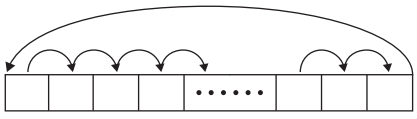
(e) Left rotate (Cyclic left shift): Rotate operations preserve all the bits being operated on. Here the bits from LSB will move one bit position to the left and MSB will placed in LSB position.



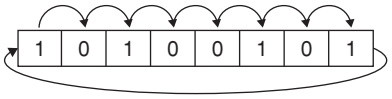
Example: $R_1 = 1010\ 0101$
Left Rotate $R_1 = 0100\ 1011$



(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.

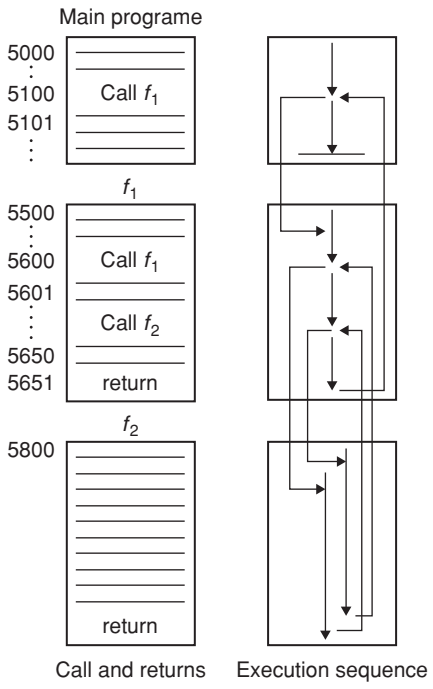
(vii) Conversion operations: These are similar to arithmetic and logical operations. May also involve special logic to perform conversion.

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

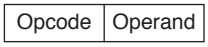
(i) Implied mode: Operands are specified implicitly in the definition of the instruction.

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

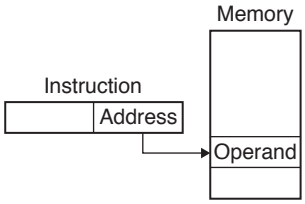


Example: Move A, 50.

- 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:

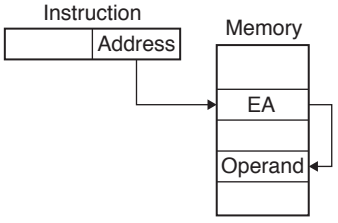
- Here the address of the operand is equal to the address part of the instruction.



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

(iv) Indirect mode:

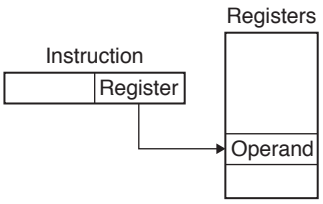
- The address field of the instruction gives the address of the operand which is stored in memory.
- The advantage of this approach is that for a word length of N, an address space of 2^N is available.
- The disadvantage is that the instruction execution requires two memory references to fetch the operand.



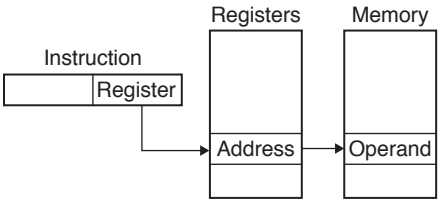
(Here EA is effective address of operand)

(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.

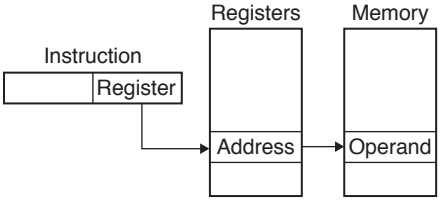


(vi) Register indirect mode: In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in memory. Address field of the instruction uses fewer bits to select a register than would have required to specify a memory address directly.



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.

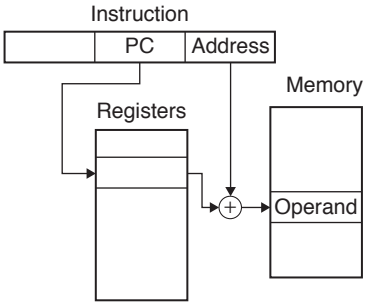
(vii) Auto increment and auto decrement mode: This is similar to register indirect mode except that the register is incremented or decremented after (or before) its value is used to access memory.



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

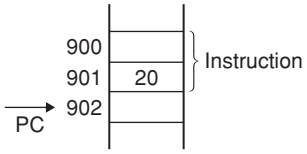
(viii) PC-relative mode: Here the content of the program counter is added to the address part of the instruction to get the effective address.

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.

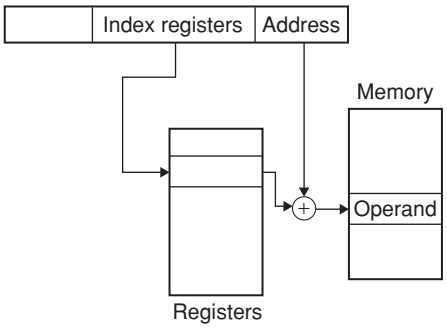


The instruction at location 900 is read memory during fetch phase and the PC will be incremented by instruction length i.e., 2. Then PC = 902.

∴ 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.

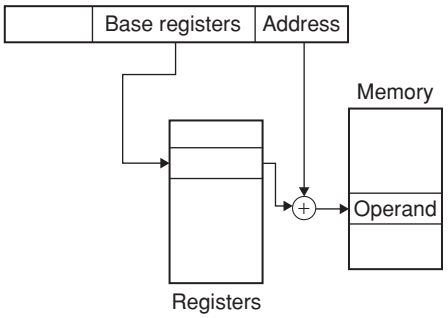


Effective Address = Index Register + Address part of instruction.

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.

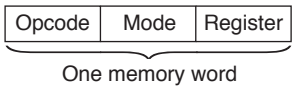
Example 3: Match the following:

	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.

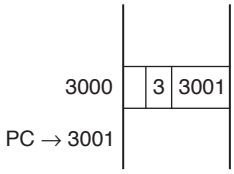
Example 4: The instruction format of a CPU is



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					