Q.2a. Give the difference between memory address, memory location and byte.

Answer: In an x86 architecture (that's what most computers use these days), memory is "byte-addressable", that is, you can access the individual bytes of memory, even though most data take up multiple bytes. Thus, when a variable takes up 2 or 4 bytes, there are several different addresses that all point to data that belongs to that variable. Your book is just telling you that pointers point to the first byte, and that the additional bytes have higher memory addresses.

Also, memory addresses are just numbers. They are usually represented as hexadecimal, as this makes more sense in most contexts, but there's no reason they cannot be expressed as decimal numbers. Remember - 0x0A and 10 are the same number, they're just written differently. Memory addresses are just integers, and can be represented any way you can represent any other integer - hex, binary, decimal, even base 13, if you like. However, most of the time, addresses are expressed in the form of 32-bit hexadecimal unsigned integers (e.g. 0xAF09128B), although the size of the address can change with the size of addressable memory (newer processors can have 40-bit, and even 64-bit address spaces).

b.Discuss the different types of operations used in instructions and instruction sequncing.

Answer: Four types of operations

1. Data transfer between memory and processor registers.
2. Arithmetic & logic operations on data
3. Program sequencing & control
4. I/O transfers.

1) Register transfer notations(RTN)

R3<–[R1]+[R2]

- Right hand side of RTN-denotes a value.
- Left hand side of RTN-name of a location.

2) Assembly language notations(ALN)

Add R1, R2, R3

- Adding contents of R1, R2 & place sum in R3.

3) Basic instruction types-4 types

- **Three address instructions**- Add A,B,C

A, B-source operands

C-destination operands

- **Two address instructions**-Add A,B

B <–[A] + [B]

- **One address instructions** –Add A

Add contents of A to accumulator & store sum back to accumulator.

- **Zero address instructions**

Instruction store operands in a structure called push down stack.

4) Instruction execution & straight line sequencing

- The processor control circuits use information in PC to fetch & execute instructions one at a time in order of increasing address.
- This is called straight line sequencing.
- Executing an instruction-2 phase procedures.
- 1st phase**–"instruction fetch"-**instruction is fetched from memory location whose address is in PC.
- This instruction is placed in instruction register in processor
- 2nd phase-**"instruction execute"-**instruction in IR is examined to determine which operation to be performed.

5) Branching

- Branch-type of instruction loads a new value into program counter.
- So processor fetches & executes instruction at this new address called "branch target"
- Conditional branch-causes a branch if a specified condition is satisfied.

- E.g. Branch>0 LOOP –conditional branch instruction .it executes only if it satisfies condition.

6) Condition codes

- Recording required information in individual bits called "condition code flags".
- These flags are grouped together in a special processor register called "condition code register" or "status register"
- Individual condition code flags-1 or 0.
- 4 commonly used flags.

1) N (negative)-set to 1 if result is –ve or else 0.

2) Z (zero)-set to 1 if result is 0, or else 0 .

3) V (overflow)-set to 1if arithmetic overflow occurs or else 0.

4) C(carry)-set to 1 if carry out results from operation or else 0

Q.3 a. What do you mean by effective address of data? List any four addressing modes. How is effective address calculated for them?

Answer: **Effective Address** :- The effective address is defined to be the memory address obtained from the computation dictated by the given addressing mode. The effective address is the address of the operand in a computational type instruction

Addressing modes:-

(i) Direct mode
(ii) Indirect mode
(iii) Register direct & register indirect mode
(iv) Immediate mode
(v) Implicit mode

(i) Direct Mode:- In this mode. The instruction includes a memory address

Ex. LDAC 5,

Data from memory location 5 is stored in accumulator.

(ii) Indirect Mode:- The address specified in the instruction is not the address of the operand. It is the address of a memory location that contains the address of the operand.

Ex. LDAC @ 5.

First it retrieve the content of memory location 5 say 10. Then the CPU goes to location 10, reads the contents of that location and loads the data into the CPU.

(iii) Register Direct & Register indirect modes:-

Register modes work the same as direct & indirect modes discussed above, except they do not specify a memory address. Instead they specify a register.

Example: LDAC R _ Content of Reg. 'R' is copied in accumulator LDAC (R) _ The content of Reg. 'R' gives the memory address whose content is to be copied in to accumulator.

(iv) Immediate Mode:- The operand specified in the instruction is not an address, it is the actual data to be used.
Example: LDAC @ 5, It moves the data value 5 to accumulator.
(v) Implicit Mode:- It does not explicitly specify an operand. The instruction implicitly specify the operand because it always applies to a specific register. Example: CLAC Clear accumulator CMC Complement accumulator.

b. What is an instruction? With example explain three, two, one, zero address instructions.

Answer:

Instruction:-
A computer has a variety of instruction code format. It is the function of the control unit within the CPU to interpret each instruction code and provide the necessary control function needed to process the instruction.
Explanation:-
The format of an instruction is appear in memory words or in a control register. The bits of the instruction are divided into groups called fields. The most common fields found in instruction formats are:
1. An operation code field that specifies the operation to be performed.
2. An address field that designates a memory address or a processor register.
3. A mode field that specifies the way the operand or the effective address is determined.
Most computer fall into one of three types of CPU organizations:
1. Single accumulator organization.
2. General register organization.
3. Stack organization.
Accumulator-type organization is the basic computer presented. All operations are performed with an implied accumulator register. The instruction format in this type of computer uses one address field.

ADD X
The instruction format in general register type organization type of computer needs three register address fields.
ADD R1, R2, R3
In the stack-organized of CPU was presented. Would have PUSH and POP instructions. PUSH x

Q.4a. Explain briefly the following: (2×4)
(i) Interrupts
(ii) Direct memory Access

Answer: i) Interrupts

There are many situations where other tasks can be performed while waiting for an I/O device to become ready. A hardware signal called an Interrupt will alert the processor when an I/O device becomes ready. Interrupt-request line is usually dedicated for this purpose.
For example, consider, COMPUTE and PRINT routines. The routine executed in response to an interrupt request is called interrupt-service routine. Transfer of control through the use of interrupts happens. The processor must inform the device that its request has been recognized by sending interrupt-acknowledge

signal. One must therefore know the difference between Interrupt Vs Subroutine. Interrupt latency is concerned with saving information in registers will increase the delay between the time an interrupt request is received and the start of execution of the interrupt-service routine.
Interrupt hardware Most computers have several I/O devices that can request an interrupt. A single interrupt request line may be used to serve n devices. Enabling and Disabling Interrupts All computers fundamentally should be able to enable and disable interruptions as desired. Again reconsider the COMPUTE and PRINT example. When a device activates the interrupt-request signal, it keeps this signal activated until it learns that the processor has accepted its request. When interrupts are enabled, the following is a typical scenario:

- The device raises an interrupt request.
- The processor interrupts the program currently being executed.
- Interrupts are disabled by changing the control bits in the processor status register (PS).
- The device is informed that its request has been recognized and deactivates the interrupt request signal.
- The action requested by the interrupt is performed by the interrupt-service routine.
- Interrupts are enabled and execution of the interrupted program is resumed.

ii) DMA

Basically for high speed I/O devices, the device interface transfer data directly to or from the memory without informing the processor. When interrupts are used, additional overhead involved with saving and restoring the program counter and other state information. To transfer large blocks of data at high speed, an alternative approach is used. A special control unit will allow transfer of a block of data directly between an external device and the main memory, without continuous intervention by the processor.
DMA controller is a control circuit that performs DMA transfers, is a part of the I/O device interface. It performs functions that normally be carried out by the processor. DMA controller must increment the memory address and keep track of the number of transfers. The operations of DMA controller must be under the control of a program executed by the processor. To initiate the transfer of block of words, the processor sends the starting address, the number of words in the block and the direction of the transfer. On receiving this information, DMA controller transfers the entire block and informs the processor by raising an interrupt signal. While a DMA transfer is taking place, the processor can be used to execute another program. After the DMA transfer is completed, the processor can return to the program that requested the transfer.
• Three registers in a DMA interface are:
• Starting address
• Word count
• Status and control flag

b.What are functions I/O interface? Explain with the help of figure.
Answer: I/O interface does the following:
1. Provides a storage buffer for at least one word of data (or one byte, in the case of byte-oriented devices)
2. Contains status flags that can be accessed by the processor to determine whether thebuffer is full (for input) or empty (for output)
3.Contains address-decoding circuitry to determine when it is being addressed by the processor
4.Generates the appropriate timing signals required by the bus control scheme
5.Performs any format conversion that may be necessary to transfer data between the bus and the I/O device, such as parallel-serial conversion in the case of a serial port

Q.5 Explain the following:- (2×8)
(i) PCI BUS
(ii) SCSI BUS

Ans (i) Page no 261 of text book
(ii) Page no 266 of text book

Q.6a. What is the difference between asynchronous and synchronous DRAM?

Answer: Conventional DRAM, of the type that has been used in PCs since the original IBM PC days, is said to be ***asynchronous*.** This refers to the fact that the memory is not synchronized to the system clock. A memory access is begun, and a certain period of time later the memory value appears on the bus. The signals are not coordinated with the system clock at all, as described in the section discussing memory access. Asynchronous memory works fine in lower-speed memory bus systems but is not nearly as suitable for use in high-speed (>66 MHz) memory systems.

A newer type of DRAM, called "**synchronous** DRAM" or "SDRAM", is synchronized to the system clock; all signals are tied to the clock so timing is much tighter and better controlled. This type of memory is much faster than asynchronous DRAM and can be used to improve the performance of the system. It is more suitable to the higher-speed memory systems of the newest PCs.

b.Discuss various types of Read only memories.

Answer:

- **Programmable read-only memory** (PROM), or **one-time programmable ROM** (OTP), can be written to or **programmed** via a special device called a **PROM programmer**. Typically, this device uses high voltages to permanently destroy or create internal links (fuses or antifuses) within the chip. Consequently, a PROM can only be programmed once.
- **Erasable programmable read-only memory** (EPROM) can be erased by exposure to strong ultraviolet light (typically for 10 minutes or longer), then rewritten with a process that again needs higher than usual voltage applied. Repeated exposure to UV light will eventually wear out an EPROM, but the **endurance** of most EPROM chips exceeds 1000 cycles of erasing and reprogramming. EPROM chip packages can often be identified by the prominent quartz "window" which allows UV light to enter. After programming, the window is typically covered with a label to prevent accidental erasure. Some EPROM chips are factory-erased before they are packaged, and include no window; these are effectively PROM.
- **Electrically erasable programmable read-only memory** (EEPROM) is based on a similar semiconductor structure to EPROM, but allows its entire contents (or selected **banks**) to be electrically erased, then rewritten electrically, so that they need not be removed from the computer (or camera, MP3 player, etc.). Writing or **flashing** an EEPROM is much slower (milliseconds per bit) than reading from a ROM or writing to a RAM (nanoseconds in both cases).
    - **Electrically alterable read-only memory** (EAROM) is a type of EEPROM that can be modified one bit at a time. Writing is a very slow process and again needs higher voltage

(usually around 12 V) than is used for read access. EAROMs are intended for applications that require infrequent and only partial rewriting. EAROM may be used as non-volatile storage for critical system setup information; in many applications, EAROM has been supplanted by CMOS RAM supplied by mains power and backed-up with a lithium battery.

- **Flash memory** (or simply **flash**) is a modern type of EEPROM invented in 1984. Flash memory can be erased and rewritten faster than ordinary EEPROM, and newer designs feature very high endurance (exceeding 1,000,000 cycles). Modern NAND flash makes efficient use of silicon chip area, resulting in individual ICs with a capacity as high as 32 GB as of 2007; this feature, along with its endurance and physical durability, has allowed NAND flash to replace magnetic in some applications (such as USB flash drives). Flash memory is sometimes called **flash ROM** or **flash EEPROM** when used as a replacement for older ROM types, but not in applications that take advantage of its ability to be modified quickly and frequently.

c. What is cache memory? Discuss direct mapping techniques

Ans Page no 317of text book

Q.7 a. What is virtual memory? Also discuss address mapping algorithm. (10)

Answer: Provides illusion of very large memory
– Sum of the memory of many jobs greater than physical memory
– Address space of each job larger than physical memory
Allows available (fast and expensive) physical memory to be well utilized.
Simplifies memory management: code and data movement, protection,
Exploits memory hierarchy to keep average access time low.
Involves at least two storage levels: main and secondary
Virtual Address -- address used by the programmer
Virtual Address Space -- collection of such addresses
Memory Address -- address in physical memory
also known as “physical address” or “real address”

Address Mapping Algorithm
If V = 1
then page is in main memory at frame address stored in table
else address located page in secondary memory
Access Control
R = Read-only, R/W = read/write, X = execute only
If kind of access not compatible with specified access rights,
then protection_violation_fault
If valid bit not set then page fault
Protection Fault: access control violation; causes trap to hardware,
or software fault handler
Page Fault: page not resident in physical memory, also causes a trap;
usually accompanied by a context switch: current process
suspended while page is fetched from secondary storage
e.g., VAX 11/780

each process sees a 4 gigabyte (232 bytes) virtual address space
1/2 for user regions, 1/2 for a system wide name space shared
by all processes.
page size is 512 bytes (too small)

b. Perform 2's complement addition of $(-5)_{10} + (-9)_{10}$ in 8 bit format.

Ans : Solution,

-5 = 2's complement of 5 =11111011

-9 = 2's complement of 9 = 11110111

Now,

11111011

+11110111

_____

1 11110010

(Note : The left most bit is the signed bit.)

Now, converting it into 2's complement we get -00001110 = -14

Q.8 a. Multiply the following pairs of signed 2's – complement numbers using the Booth algorithm: A = 001111 and B = 001111

b. Discuss the Arithmetic operations on floating point numbers

Answer:

Arithmetic Operations on Floating-Point Numbers:

The rules apply to the single-precision IEEE standard format. These rules specify only the major steps needed to perform the four operations. Intermediate results for both mantissas and exponents might require more than 24 and 8 bits, respectively & overflow or an underflow may occur. These and other aspects of the operations must be carefully considered in designing an arithmetic unit that meets the standard. If their exponents differ, the mantissas of floating-point numbers must be shifted with respect to each other before they are added or subtracted. Consider a decimal example in which we wish to add 2.9400 x 102 to 4.3100 x 104. We rewrite 2.9400 x 102 as 0.0294 x 104 and then perform addition of the mantissas to get 4.3394 x 104. The rule for addition and subtraction can be stated as follows:

Add/Subtract Rule
The steps in addition (FA) or subtraction (FS) of floating-point numbers (s1, *e*ˆ , f1) fad {s2, *e*ˆ 2, f2) are as follows.
1. Unpack sign, exponent, and fraction fields. Handle special operands such as zero, infinity, or NaN(not a number).
2. Shift the significand of the number with the smaller exponent right by *e*1□*e*2 bits.
3. Set the result exponent er to max(e1,e2).
4. If the instruction is FA and s1= s2 or if the instruction is FS and s1 ≠ s2 then add the significands; otherwise subtract them.
5. Count the number z of leading zeros. A carry can make z = -1. Shift the result significand left z bits or right 1 bit if z = -1.
6. Round the result significand, and shift right and adjust z if there is rounding overflow, which is a carry-out of the leftmost digit upon rounding.
7. Adjust the result exponent by er = er - z, check for overflow or underflow, and pack the result sign, biased exponent, and fraction bits into the result word.

Q.9 a. With the help of figure, explain multiple-bus organization.

Ans Page no. 423-424 of text book

b. Show, how a microprogrammed control unit works? Discuss its advantages over conventional control unit.

Answer: Microprogrammed Control unit uses a fast memory (called Control ROM) to store microinstructions. Each microinstruction generates control signals to cause one or more micro-operations. A set of microinstructions make a microprogram, which corresponds to an instruction, and is used to FETCH, DECODE and execute the instructions.
Data in a microinstruction of the Control ROM may be organized as shown below
Internal CPU System Bus Jump Condition Next microinstruction
Control Signals Control Signals Bits Address
To execute an instruction, following steps are involved
1. Requisite control signals are turned ON
2. If the jump condition is FALSE, next microinstruction in sequence is executed
3. If jump condition is TRUE, microprogram jumps to address specified in the last field.

TEXT BOOK

1. **Computer Organization, Carl Hamacher, Zvonko Vranesic, Safwat Zaky, 5th Edition, TMH, 2002**