MCS-012

1. (Covers Block 1)

(a)

How can you represent a negative integer in a computer system? If 8 bits (including one sign bit) are to be used to represent integers in binary 2"s complement notation, then what are the possible minimum and maximum numbers that can be represented? Perform the following arithmetic operations using signed 2"s complement, 8 bit representation. (Please note that the numbers given below are in decimal notation)

i) Subtract 198 from –98 ii) Add 124 and 142 Please indicate the overflow if it is occurs. How have you identified the overflow?

(2 Marks)

Ans:

Complements are used to represent to negative numbers in digital computers. If base of number system is 'r' then complements will be r's complement and (r-1)'s complement.

Example:

- 5 is written in 8 bit register as binary: 00000101
- -5 is written in 8 bit register as binary: 11111010(1's Complements)
- -5 is written in 8 bit register as binary: 11111010+1=11111011 (2's Complements).

(i)

98➔ 0 1 1 0 0 0 1 0

-98➔ 0 1 0 0 1 1 1 0 1

198➔01 1 0 0 0 1 1 0 overflow

(ii)

124➔ 0 1 1 1 1 1 0 0

142➔ 1 0 0 0 1 1 1 0

---

1 0 0 0 0 1 0 1 0 overflow

If carry into sign bit is not equal to carry out sign bit then overflow must occurs.

(b)

Perform the following conversion of numbers:

(i) Decimal (2050)10 to hexadecimal

(ii) (ii) Hexadecimal (19BACDFE)H into Octal.

(iii) (iii) ASCII string "AssignMenT" into UTF 16

(iv) (iv) Octal (547561)O into Hexadecimal

(2 Marks)

Ans:

(i) *802*
(ii) 3156546776
(iii) 0061 006C 0067 006F 0072 0069 0074 0068 006D
(iv) 2CF71

(c)

A combinational circuit is to be designed that counts the number of occurrences of 1 bits in a 4 bit input, however, an input 1111 is an invalid input for the circuit and output in such a case will be 00. One valid input for such circuit may be 1110 having the output 11; another valid input may be 1010 with the output 10. Draw the truth table for the circuit. Use the Karnaugh's map to design the circuit and draw it using AND, OR and NOT gates.

(4 Marks)

Ans:

(d)

What is parity bit? Explain how Single Error Correcting (SEC) code uses parity bits. If an 8 bit data 10101010 on transmission is received as 10111010, then how the SEC code will detect and correct this error.

Ans:

The equation for SEC code is

2^i-1 > =N+i

Where i= Number of bits in SEC code and

N= Number of bits in data word.=?

the equation is

2^i -1 > =8+ i

at i=3 2^3 – 1 >= 8 + 3

7 >= 11 (Not True)

Put i = 4

2^4 -1 > =8+4

15 > = 12 satisfied .

True the condition is satisfied.

Although, this condition will be true for i > 4 also but we want to use only minimum essential correction bits which are 4.

For SEC-DED code we require an additional bit as overall parity. Therefore, the SECDED code will be of 5 bits.

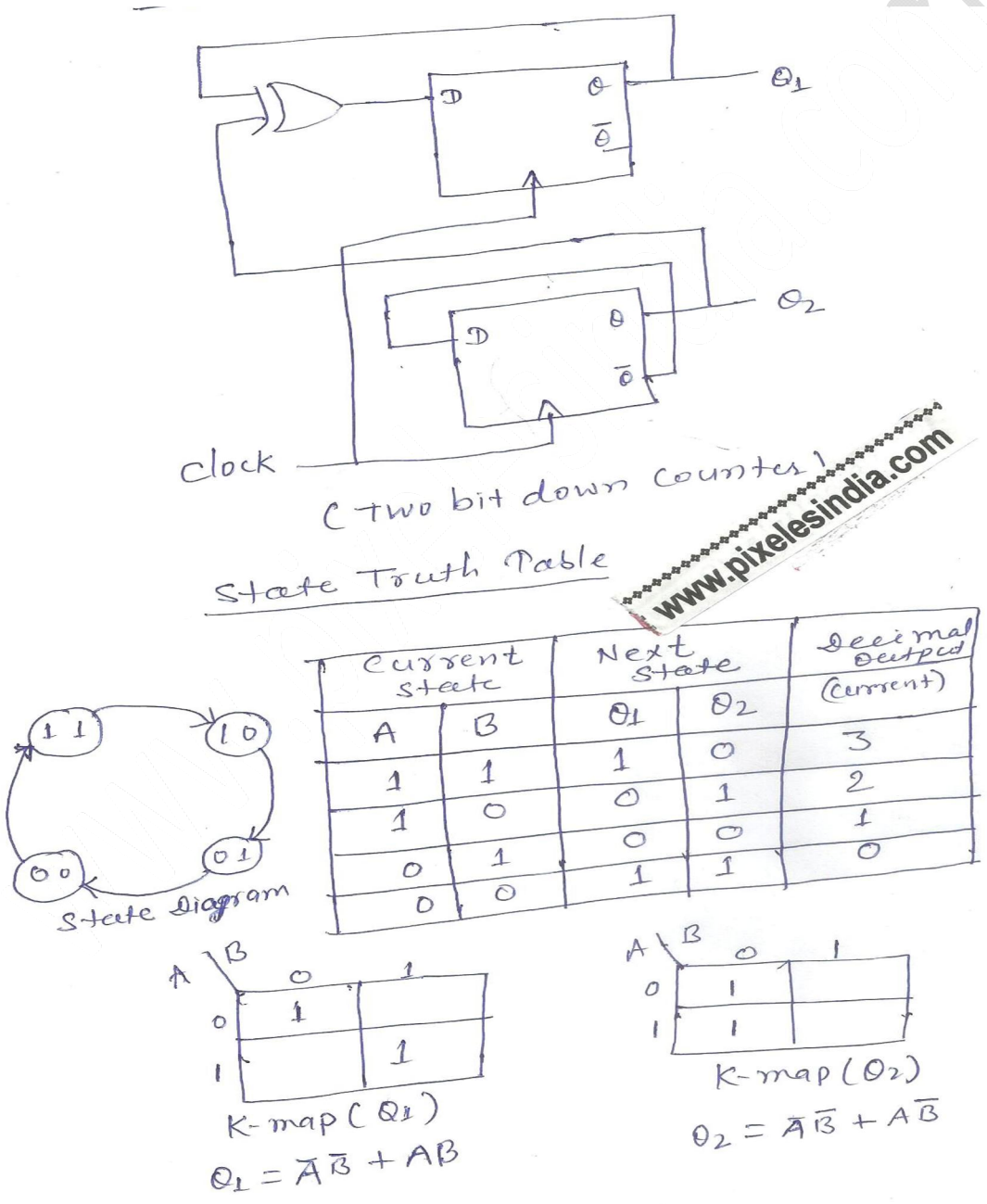
The 8-bit Sent Data = 0110 1101 The 8-bit Sent Data = 0100 1101 Check with parity bits, before that Create a new parity bit with new data P1= D1 D2 D4 D5 D7 = 01010 even parity will be 0 – correct P2= D1 D3 D4 D6 D7= 00010 even parity will be 1 - incorrect P3= D2 D3 D4 D8 = 1001 even parity will be 0 - incorrect P4= D5 D6 D7 D8 = 1101 even parity will be 1 – correct After Checking We find that common data bit numbers are D3 & D4 But D4 is also present in P1that means error has been

occurred in D3 only. By this we dectect error in D3 and Correct it by replacing it by 0 to 1.

(e)

Design a two bit counter (a sequential circuit) that counts in reverse order, i.e. from 11 to 00. Thus, the counter states are 11, 10, 01, 00, 11, 10, 01, 00, 11 … You should show the state table, state diagram, the k-map for circuit design and logic diagram of the resultant design using D flip-flop or J-K flip flop. , *(4 Marks)*

Ans:



Current state		Next state		Decimal Output (current)
A	B	Q_1	Q_2	
1	1	1	0	3
1	0	0	1	2
0	1	0	0	1
0	0	1	1	0

(f) , Explain the single precision floating point IEEE 754 representation. Give the number ranges that can be represented by this representation. Also, represent the number (356.122)10 using IEEE 754 single precision as well as double precision representations. Is the representation of the said number exactly same in the two representations? Explain your answer. , *(4 Marks)*

IEEE floating point numbers have three basic components: the sign, the exponent, and the mantissa.32 bits floating point is called single precision and 64 bits floating point is known as double precision. It can be represented as following:-

	Sign	Exponent	Fraction	Bias
Single Precision	1 [31]	8 [30-23]	23 [22-00]	127
Double Precision	1 [63]	11 [62-52]	52 [51-00]	1023

The Sign Bit:- it describes sign of number. 0 denotes a positive number; 1 denotes a negative number.

The Exponent

The exponent field needs to represent both positive and negative exponents. To do this, a *bias* is added to the actual exponent in order to get the stored exponent. For IEEE single-precision floats, this bias value is 127. For double precision, it has a bias of 1023.

The Mantissa

The *mantissa*, also known as the *significant*, represents the precision bits of the number. It is composed of an implicit leading bit and the fraction bits.

Binary of 356.122➔ 101100100.000
Normalized ➔1.01100100000 X 2^{-8}

Mantissa = 0110010000
Exponent=126+ -8=118➔ 01110110

2. (Covers Block 2)

(a) , How is the word size of RAM and its capacity related to number of addressing bits? A RAM has a capacity of 1M words having the word size of 32 bits and supports byte addressing.
(i) How many data input and output lines does this RAM need? Explain your answer.
(ii) How many address lines will be needed for this RAM?
, *(2 Marks)*

Ans:

Input/output line =4.

*Address Line=8*2014➔8192➔2^{13} ➔13 address lines*

(b) , A hypothetical computer has 16 MB RAM and has a word size of 32 bits. It has cache memory having 32 blocks having a block size of 64 bits. Show how the main memory address 100110001111101110111100 will be mapped to a cache address, if (i) Direct cache mapping is used (ii) Associative cache mapping is used

(iii) Two way set associative cache mapping is used.

, *(4 Marks)*

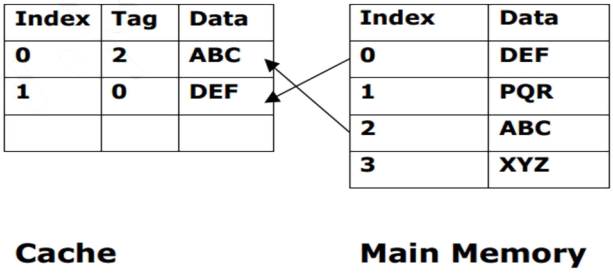
Ans:

In this technique, Cache memory and main memory is divided into blocks. The Block '*k*' of main memory maps into block ***k* modulo m** of the cache, where *m* is the total number of blocks in cache. Each block mapped to exactly 1 cache location.

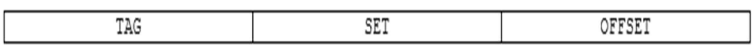
Cache location = (block address) MOD (# blocks in cache)

A given memory block can be mapped into one and only cache line. Here is an example of mapping Cache line Main memory block if total number of block in cache is 8.

Cache line	Main memory block
0	0, 8, 16, 24, ... 8n
1	1, 9, 17. 25, ... 8n+1
2	2, 10, 18, 26, ... 8n+2
3	3, 11, 19, 27, ... 8n+3



An address Consists of following filed:-



Example: A cache is direct-mapped and has 64 KB data. Each block contains 32 bytes. The address is 32 bits wide. What are the sizes of the tag, index, and block offset fields?

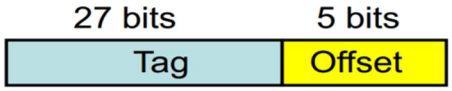
- **Bits of offset = 5 (since each block contains 2^5 bytes=32)**
- **blocks in cache = 64×1024 / 32 = 2048 blocks**

- **bits in index field = 11 (since there are 2^11 blocks=2048)**
- **bits in tag field = 32 - 5 - 11 = 16**

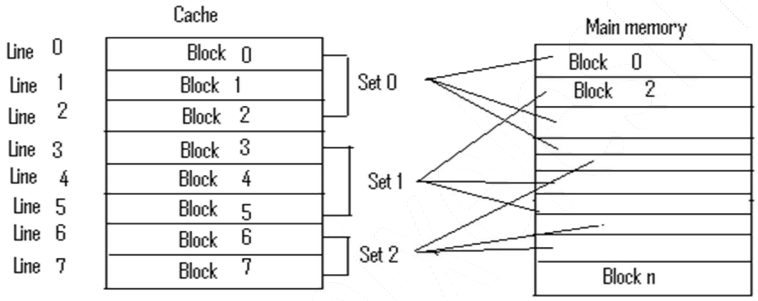
Associated mapping:

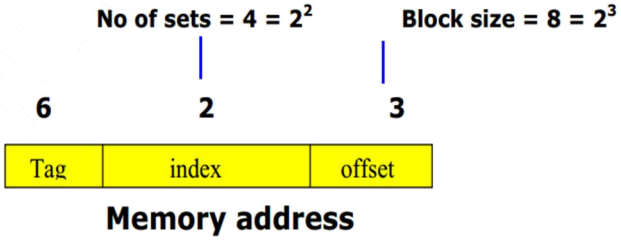
In this mapping scheme, any block of main memory can be mapped on to any location cache memory.

- A main memory block can load into any block of cache
- Memory address is interpreted as tag and word
- Tag uniquely identifies block of memory
- Every cache block's tag is examined for a match
- Cache searching gets expensive



Set-associated mapping: - It is a combination of direct mapping and associated mapping. In this mapping scheme set of blocks are identified by using direct mapping scheme and lines (cache block) within each set are identified by associative cache. Cache contains of 2^n sets of m lines (cache block)





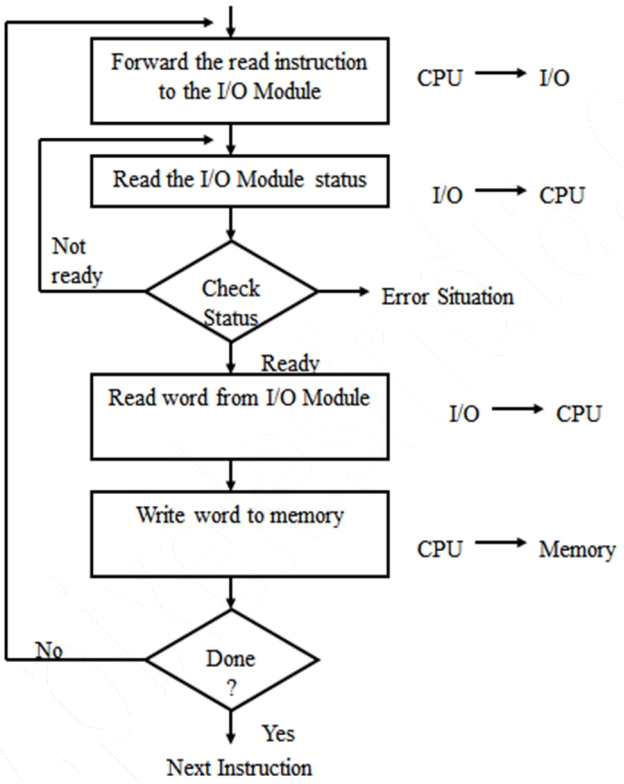
(c) , Explain the basic features of the three I/O techniques (Programmed I/O, Interrupt driven I/O and DMA) with the help of diagrams. A computer is to be designed for an environment requiring frequent disk transfers. Which of the three I/O techniques is most suitable for this computer? Justify your answer. , ***(4 Marks)***

Ans:

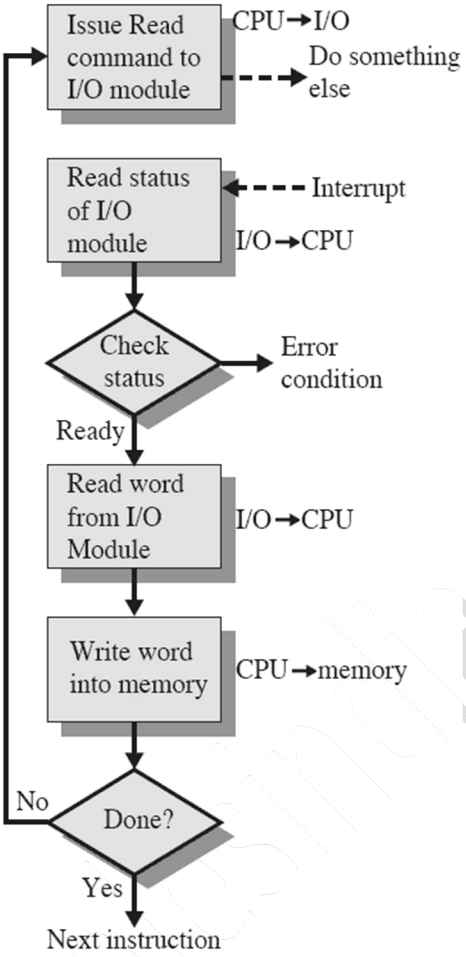
Programmed Input/output: -

Programmed Input/output (PIO) is a way of moving data between device s in a computer in which all data must pass through the processor. It provides:-

- Transfer of data from I/O device to the CPU registers
- Transfer of data from CPU registers to memory.
  In a programmed I/O method the responsibility of CPU is to constantly check the status of the I/O device to check whether it has become free or not. Thus, Programmed I/O is a very time consuming method where CPU wastes lot of time of checking and verifying the status of an I/O device. In programmed I/O, the I/O operations are completely controlled by the CPU.



INTERRUPT DRIVEN I/O:-The problem with programmed I/O is that the processor has to wait a long time for the I/O module of concern to be ready for either reception or transmission of data. The solution to this problem is to provide an interrupt mechanism. In this approach the processor issues an I/O command to a module and then go on to do some other useful work? The I/O module with then interrupt the processor to request service when it is ready to exchange data with the processor.

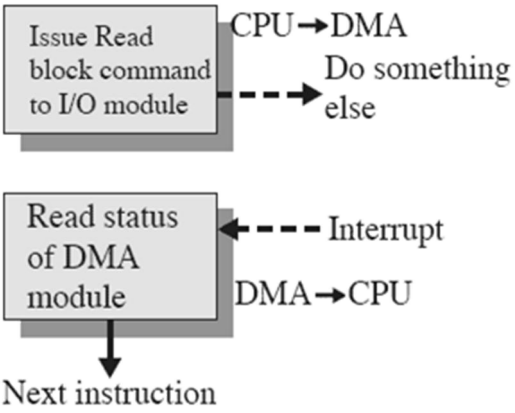


(b) Interrupt-driven I/O

DMA (Direct memory access)

Direct memory access (DMA) is a method that allows an input/output (I/O) device to send or receive data directly to or from the main memory, without passing through CPU to speed up memory operations. The process is managed by a chip known as a DMA controller

Direct memory access (DMA). Module is required when large amount of data is to be transferred. DMA transfers the requested block bytes by byte directly to the memory without CPU interaction, after completion the transfer (request) DMA send a signal to CPU. Thus we can say DMA module perform the task requested by CPU. This type of data transfer is called ***direct memory access (DMA)***.



(c) Direct memory access

(d) , Consider a file having name *mca.txt* and is of size 20 K. You have a disk having 32 tracks, each track having 16 sectors with each sector being 1K. Assume that disk has three free - continuous clusters of 8 sectors each. How can this file be given the space on the disk? Show the content of FAT after the space allocation to the file. You may make suitable assumptions. You may assume the cluster size as 4 sectors.
Ans:

Layout

First the boot sector (at relative address 0), and possibly other stuff. Together these are the Reserved Sectors. Usually the boot sector is the only reserved sector.

Then the FATs (following the reserved sectors; the number of reserved sectors is given in the boot sector, bytes 14-15; the length of a sector is found in the boot sector, bytes 11-12).

Then the Root Directory (following the FATs; the number of FATs is given in the boot sector, byte 16; each FAT has a number of sectors given in the boot sector, bytes 22-23).

Finally the Data Area (following the root directory; the number of root directory entries is given in the boot sector, bytes 17-18, and each directory entry takes 32 bytes; space is rounded up to entire sectors).

Boot sector

The first sector (512 bytes) of a FAT filesystem is the *boot sector*. In Unix-like terminology this would be called the superblock. It contains some general information.

First an explicit example (of the boot sector of a DRDOS boot floppy).

```
0000000 eb 3f 90 49 42 4d 20 20 33 2e 33 00 02 01 01 00
0000020 02 e0 00 40 0b f0 09 00 12 00 02 00 00 00 00 00
0000040 00 00 00 00 00 00 00 00 00 00 70 00 ff ff 49 42
0000060 4d 42 49 4f 20 20 43 4f 4d 00 50 00 00 08 00 18
...
```

(See here for the complete sector. And also a MSDOS example)

The 2-byte numbers are stored little endian (low order byte first).

Here the FAT12 version, that is also the common part of the FAT12, FAT16 and FAT32 boot sectors. See further below.

Bytes Content

0-2 Jump to bootstrap (E.g. eb 3c 90; on i86: JMP 003E NOP.
One finds either eb xx 90, or e9 xx xx.
The position of the bootstrap varies.)

3-10 OEM name/version (E.g. "IBM 3.3", "IBM 20.0", "MSDOS5.0", "MSWIN4.0".
Various format utilities leave their own name, like "CH-FOR18".
Sometimes just garbage. Microsoft recommends "MSWIN4.1".)
/* BIOS Parameter Block starts here */

11-12 Number of bytes per sector (512)
Must be one of 512, 1024, 2048, 4096.

13 Number of sectors per cluster (1)
Must be one of 1, 2, 4, 8, 16, 32, 64, 128.
A cluster should have at most 32768 bytes. In rare cases 65536 is OK.

14-15 Number of reserved sectors (1)
FAT12 and FAT16 use 1. FAT32 uses 32.

16 Number of FAT copies (2)

17-18 Number of root directory entries (224)
0 for FAT32. 512 is recommended for FAT16.

19-20 Total number of sectors in the filesystem (2880)
(in case the partition is not FAT32 and smaller than 32 MB)

21 Media descriptor type (f0: 1.4 MB floppy, f8: hard disk; see below)

22-23 Number of sectors per FAT (9)
0 for FAT32.

24-25 Number of sectors per track (12)

26-27 Number of heads (2, for a double-sided diskette)

28-29 Number of hidden sectors (0)
Hidden sectors are sectors preceding the partition.
/* BIOS Parameter Block ends here */
30-509 Bootstrap
510-511 Signature 55 aa

The signature is found at offset 510-511. This will be the end of the sector only in case the sector size is 512.

Media descriptor byte

The ancient media descriptor type codes are:

For 8" floppies:
fc, fd, fe - Various interesting formats

For 5.25" floppies:

Value	DOS version	Capacity	sides	tracks	sectors/track
ff	1.1	320 KB	2	40	8
fe	1.0	160 KB	1	40	8
fd	2.0	360 KB	2	40	9
fc	2.0	180 KB	1	40	9
fb		640 KB	2	80	8
fa		320 KB	1	80	8
f9	3.0	1200 KB	2	80	15

For 3.5" floppies:

Value	DOS version	Capacity	sides	tracks	sectors/track
fb		640 KB	2	80	8
fa		320 KB	1	80	8
f9	3.2	720 KB	2	80	9
f0	3.3	1440 KB	2	80	18
f0		2880 KB	2	80	36

For RAMdisks:
fa

For hard disks:

Value	DOS version
f8	2.0

This code is also found in the first byte of the FAT.

IBM defined the media descriptor byte as 11111red, where r is removable, e is eight sectors/track, d is double sided.

Cluster size

The default number of sectors per cluster for floppies (with FAT12) is

Drive size	Secs/cluster	Cluster size
360 KB	2	1 KiB
720 KB	2	1 KiB
1.2 MB	1	512 bytes
1.44 MB	1	512 bytes
2.88 MB	2	1 KiB

(e) Explain the following giving their uses and advantages/disadvantages. (Word limit for answer of each part is 50 words ONLY)

(i) DVD
(ii) Monitor Resolution
(iii) Non-impact printers
(iv) Scan codes
(v) Graphics accelerators
(vi) SCSI

(a)

A hypothetical machine has 64 general purpose registers of 64 bits each. The machine has 4G word of RAM (assume that each word is of 64 bits and memory is word addressable). The instructions of machine are of fixed format and are 64 bit long. Instructions of the machine consist of operation code, addressing mode specification, one register operand and one memory operand. The machine uses 2 bits to specify addressing mode as given below:
Addressing mode bits , Register Operand , Memory Operand
00 , Direct , Direct
01 , Direct , Immediate data
10 , Register Indirect , Direct
11 , Register Indirect , Immediate data

Assuming that the instructions are first fetched to Instruction Register (IR) and the two operands are transferred to AC and DR registers respectively, and result of operation is stored in the AC register; write and explain the sequence of micro-operations that are required for fetch and execute cycles of an ADD instruction having addressing mode bits as 01. Make and state suitable assumptions, if any.

, *(6 Marks)*

Ans:

Addressing modes	**Example Instruction**	**Meaning**	**When used**
Register	Add R4,R3	R4 <- R4 + R3	When a value is in a register
Immediate	Add R4, #3	R4 <- R4 + 3	For constants
Displacement	Add R4, 100(R1)	R4 <- R4 + M[100+R1]	Accessing local variables
Register deffered	Add R4,(R1)	R4 <- R4 + M[R1]	Accessing using a pointer or a computed address
Indexed	Add R3, (R1 + R2)	R3 <- R3 + M[R1+R2]	Useful in array addressing: R1 - base of array R2 - index amount
Direct	Add R1, (1001)	R1 <- R1 + M[1001]	Useful in accessing static data
Memory deferred	Add R1, @(R3)	R1 <- R1 + M[M[R3]]	If R3 is the address of a pointer *p*, then mode yields **p*
Auto-increment	Add R1, (R2)+	R1 <- R1 +M[R2] R2 <- R2 + *d*	Useful for stepping through arrays in a loop. R2 - start of array *d* - size of an element
Auto-decrement	Add R1,-(R2)	R2 <-R2-*d* R1 <- R1 + M[R2]	Same as autoincrement. Both can also be used to implement a stack as push and pop
Scaled	Add R1, 100(R2)[R3]	R1<- R1+M[100+R2+R3**d*]	Used to index arrays. May be applied to any base addressing mode in some machines.

(b) , Assume that you have a machine as shown in section 3.2.2 of Block 3 having the micro-operations as given in Figure 10 on page 62 of Block 3. Consider that R1 and R2 both are 8 bit registers and contains 11010011 and 10000111 respectively. What will be the values of select inputs, carry-in input and result of operation (including carry out bit) if the following micro-operations are performed? (For each micro-operation you may assume the initial value of R1 and R2 as given above)

(i) Subtract with borrow R2 from R1

(ii) Exclusive OR of R1 and R2

(iii) Shift left R1 twice

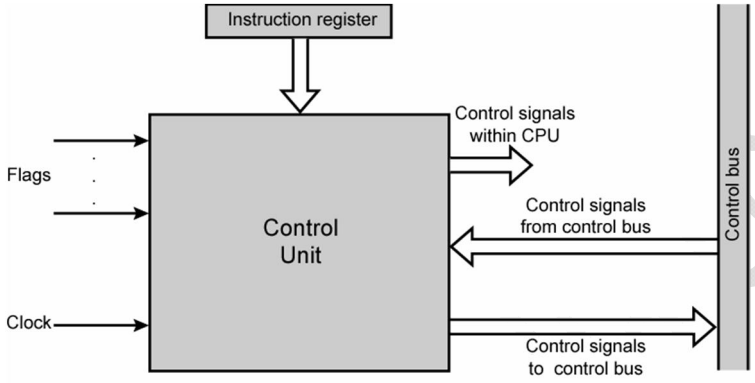
(iv) Increment R1

, *(2 Marks)*

(c) , What are the functions of a control unit? Compare and contrast the functioning of hardwired control unit to that of micro-programmed control Unit. , *(3 Marks)*

Ans:

Control Unit



Control unit controls the input and output execution.

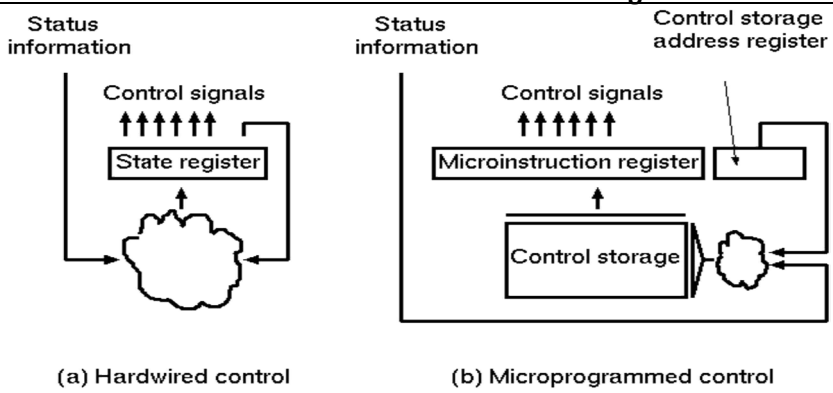
Control unit operations

- **Master clock signal:** -Clock signal to be set 0 or 1 for the micro operation that means it indicates the clock signal and count the time how many ns taken by micro operation.
- **Instruction register:-** Instruction register of CU determines the addressing mode of bits (instruction) that means operational code always need address to perform the micro operation.
- **Flags:** - It is used to determine the status CPU.
- **Control signals from control bus:** - CU receives the control signal from the control bus that means outside of CPU.
- **Control signal within CPU:** - That means such type of control signal is used for micro operations and transfer data from one register to another register.

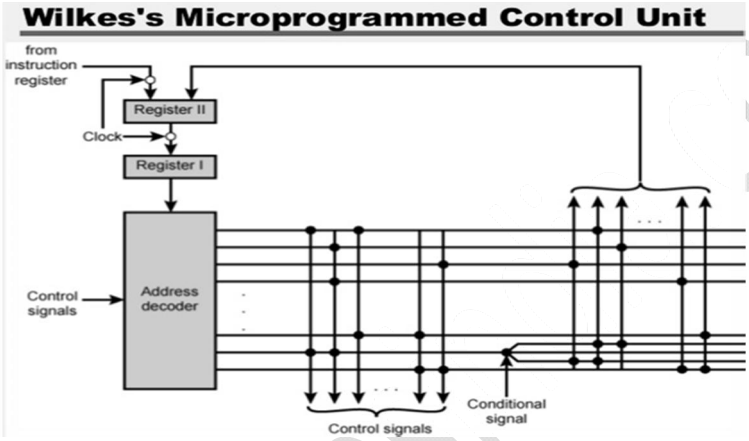
Output control operation

- **Control signal in CPU:** - Such type of signal is used to transfer of the data from one register to another register means output register.
- **Control signal to control bus:** - Such type of control signal transfer data form CPU registers to main memory input/output device.

A **micro programmed control unit** is a relatively simple logic circuit that is capable of sequencing through microinstructions and generating **control** signals to execute each microinstruction. The concept of **micro program** is similar to computer program. It was introduced by Wilkes, so, it is called Wilkes micro programmed Unit.



(a) Hardwired control (b) Microprogrammed control



Hardwired control

Hardwired control units are implemented through use of sequential logic units, featuring a finite number of gates that can generate specific results based on the instructions that were used to invoke those responses. Hardwired control units are generally faster than micro programmed designs.

In the hardwired organization, the control logic is implemented with gates, flip-flops, decoders, and other digital circuits. It has the advantage that it can be optimized to produce a fast mode of operation.

(d) , Explain the differences between the RISC and CISC machines. Also explain the differences in the pipelining of these two types of machines. , *(2 Marks)*

Ans:

CISC	RISC
Emphasis on hardware	Emphasis on software
Includes multi-clock complex instructions	Single-clock, reduced instruction only
Memory-to-memory: "LOAD" and "STORE" incorporated in instructions	Register to register: "LOAD" and "STORE" are independent instructions

Small code sizes, high cycles per second	Low cycles per second, large code sizes
Transistors used for storing complex instructions	Spends more transistors on memory registers
Large number of instructions – from 120 to 350	Relatively fewer instructions - less than 100
Micro programmed Control Unit.	Hardwired Control Unit.
Variable-length instruction formats.	Fixed-length instructions usually 32 bits, easy to decode instruction format

(e) , Assume that a RISC machine has 256 registers out of which 48 registers are reserved for the Global variables and 64 for Instruction related tasks. This machine has been designed to have 16 registers for storing four input parameters, four output parameters and eight local variables for function call. Explain with the help of a diagram, how the overlapped register window can be implemented in this machine for a function/procedure calls. You must explain how the parameters will be passed when a function calls another function. , ***(3 Marks)***

Ans:

Incoming Parameter Registers	Outgoing Parameter Registers	No. of Local Registers
1	1	22
2	2	20
3	3	18
4	4	16
5	5	14
6	6	12
7	7	10
8	8	8
9	9	6
10	10	4
11	11	2
12	12	0

Assume that the number of incoming parameters is equal to the number of outgoing parameters. Therefore, Number of locals = 24 –(2 × Number of incoming parameters) Return address is also counted as a parameter, therefore, number of incoming parameters is more than or equal to 1 or in other terms the possible combination,
are:

4. (Covers Block 4)

(a) , Write a program in 8086 assembly Language (with proper comments) that accepts a string of four characters entered using the keyboard and checks if all the entered characters are decimal digits. In case all the characters are decimal digits then it converts the entered string into equivalent binary number. Make suitable assumptions, if any.

Write a program in 8086 assembly Language (with proper comments) that passes a parameter containing a lower case alphabet to a near procedure named TOUPCASE, which converts it to upper case and returns it to the calling assembly program. Make suitable assumptions, if any. , (7 Marks)

Ans:

```
DATA SEGMENT
BCD   DW     4567h
HEX   DW     ? ;
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START: MOV AX, DATA
        MOV   DS, AX
        MOV   AX, BCD
        MOV   BX, AX
        MOV   AL, AH
        MOV   BH, BL
        MOV CL, 04
        ROR AH, CL
        ROR BH, CL ;
        AND AX, 0F0FH
        AND BX, 0F0FH
        MOV CX, AX
        MOV AX, 0000H

        MOV AL, CH
        MOV DI, THOU
        MUL DI
        MOV DH, 00H
        MOV DL, BL
        ADD DX, AX
        MOV AX, 0064h
        MUL CL
        ADD DX, AX
        MOV AX, 000Ah
        MUL BH
        ADD DX, AX
        MOV HEX, DX
```

```
        MOV AX, 4C00h
INT 21h
CODE ENDS
END START
```

(c) , Explain the following in the context of 8086 Microprocessor (i) Use of segment and segment registers (ii) Interrupt vector table and its use (iii) Indirect Addressing Modes of 8086 microprocessor

SEGMENT Directive: The segment directive defines the logical segment to which subsequent instructions or data allocations statement belong. It also gives a segment name to the base of that segment. The address of every element in a 8086 assembly program must be represented in segment - relative format.

An interrupt vector is the memory location of an interrupt handler, which prioritizes interrupts and saves them in a queue if more than one interrupt is waiting to be handled.

An interrupt is a signal from a device attached to a computer, or from a program within the computer, that tells the OS (operating system) to stop and decide what to do next. When an interrupt is generated, the OS saves its execution state by means of a context switch, a procedure that a computer processor follows to change from one task to another while ensuring that the tasks do not conflict. Once the OS has saved the execution state, it starts to execute the interrupt handler at the interrupt vector.

Indirect addressing: - In this scheme, operands require two memory references, so it is allocated additional memory space. The drawback of this scheme is that it requires two memory references to fetch the actual operand. The first memory reference is to fetch the actual address of the operand from the memory and the second to fetch the actual operand using that address, it is represented by

EA = (A)
D = (EA)