काशी हिन्दू विश्वविद्यालय BANARAS HINDU UNIVERSITY

B.Sc. (HONS.) IV Semester Examination 2020-21

COMPUTER SCIENCE

CS-104: Computer Organization and Architecture

Presented By:-

Anjali Gupta (20229STA012)

Goldi (20229PHY005)

Ishita Pathak (20229MAT011)

Laxmi Singh (20229CMP003)

Prity (20229STA016)

Ritu Kumari (20229MAT004)

Guided By:- Dr. Sarvesh Pandey

MEMORY MAPPING

Q.No.1. A direct mapped cache memory of 1 MB has a block size of 256 bytes. The cache has an access time of 3 ns and a hit rate of 94%. During a cache miss, it takes 20 ns to bring the first word of a block from the main memory, while each subsequent word takes 5 ns. The word size is 64 bits. The average memory access time in ns (round off to 1 decimal place) is?

Ans. Given, Word size= 64 bits= 8 bytes

And, Block size= 256 bytes

Therefore, No. of words per block will be= 256 / 8 = 32

According to question, first word takes 20ns and rest (31) each subsequent words take 5ns each to fetch a word from main-memory to cache.

Hence, Tag = (0.94 × 3) + (1 – 0.94) [3 + (20 + (31 × 5))]= 13.5 (in ns)

Q.No.2. A 4-way set-associative cache memory unit with a capacity of 16 KB is built using a block size of 8 words. The word length is 32 bits. The size of the physical address space is 4 GB. The no. of bits for the TAG field is?

Ans. In a k-way set associate mapping, cache memory is divided into sets, each of size k blocks.

Size of Cache memory= 16 KB

As it is 4-way set associative, K = 4

Block size, B = 8 words

The word length is 32 bits.

Size of Physical address space = 4 GB

No. of blocks in Cache Memory (N) = (size of cache memory / size of a block)

= (16*1024 bytes / 8*4 bytes) = 512 (as 1 word = 4 bytes)

No. of sets (S) = (No. of blocks in cache memory / No. of blocks in a set) = N/K=512/4 = 128

Now, size of physical address = 4GB = 4*(2^30) Bytes = 2^32 Bytes

These physical adresses are divided equally among the sets.

Hence, each set can access ((2^32)/128) bytes = 2^25 bytes = 2^23 words = 2^20 blocks

So, each set can access total of 2^20 blocks. So to identify these 2^20 blocks, each

set needs TAG bits of length 20 bits.

Q.No.3. Consider a machine with a byte addressable main memory of 220 bytes, block size of 16 bytes and a direct mapped cache having 212 cache lines. Let the addresses of two consecutive bytes in main memory be (E201F)16 and (E2020)16. What are the tag and cache line address (in hex) for main memory address (E201F)16?

Ans. Block Size = 16 bytes

Block Offset = 4

No. of sets or cache lines = 212

No. of index bits = 12

Size of main memory = 220

No. of tag bits = 20 - 12 - 4 = 4

Let us consider the hex address E201F

Tag lines = First 4 bits = E (in hex)

Cache lines = Next 12 bits = 201 (In Hex)

Q.No.4. A cache memory unit with capacity of N words and block size of B words is to be designed. If it is designed as direct mapped cache, the length of the TAG field is 10 bits. If the cache unit is now designed as a 16-way set-associative cache, the length of the TAG field is __ bits?

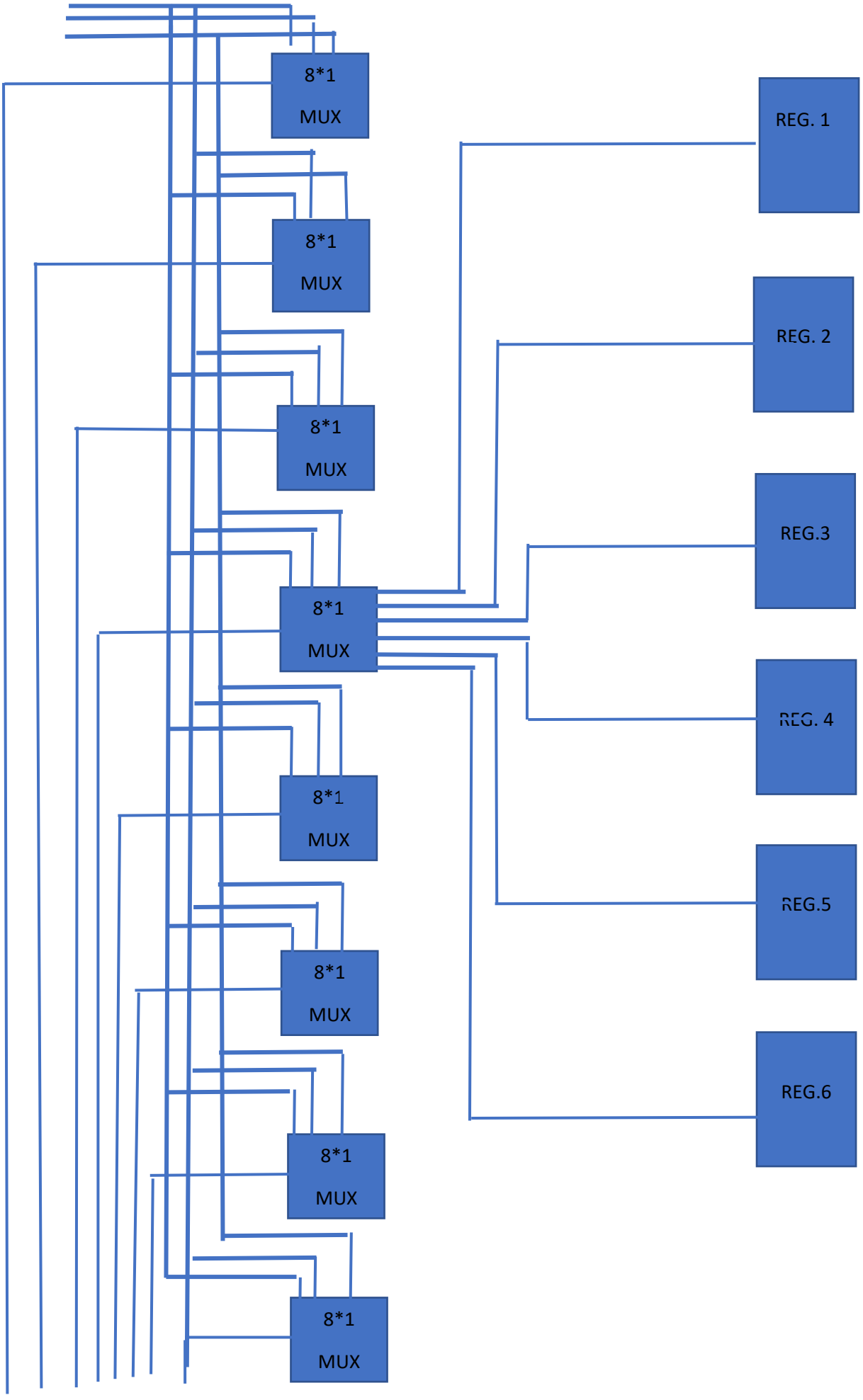
Ans. Type of mapping is direct map; for this direct map, 10 bits are required in its Tag.

It is updated to 16 way set Associative map then new tag field size = 10 + log216 = 14 bits, because for k way set associative map design, log2k bits are additionally required to the number of bits in tag field for direct map design.

QUESTION PAPER

Q.No.1.(a) Draw a diagram of a bus system for 6 registers of 8 bits each using multiplexer.

Ans.


8*1
MUX
8*1
MUX
8*1
MUX
8*1
MUX
8*1
MUX
8*1
MUX
8*1
MUX
8*1
MUX
REG. 1
REG. 2
REG.3
REG. 4
REG.5
REG.6

(b) What is the difference between logical, circular and arithmetic shift? Starting from an initial value of R = 11000101, determine the sequence of binary values in R after a logical shift-left , followed by a circular shift- right, followed by a logical shift -right and circular shift-right . Give the decimal equivalent of the final value of R.

Ans. **LOGICAL SHIFT-** It transfers the zero through the serial input.

Left Logical Shift- In this shift one position moves each bit to the left one by one. The empty least significant bit (LSB) is filled with zero and the most significant bit (MSB) is rejected.

Right Logical Shift- In this one position moves each bit to the right one by one and the least significant bit (LSB) is rejected and the empty MSB is filled with zero.

CIRCULAR SHIFT- It shifts the bits of the register around the two ends without loss of information. This is accomplished by connecting the serial input.

It also have both sides circular shift :

1. Left Circular Shift

2. Right Circular Shift

ARITHMETIC SHIFT- This micro-operation shifts a signed binary no. to the left or to the right position. In an arithmetic shift left, it multiplies a singled binary no. by 2 and in an arithmetic shift right, it divides the no. by 2.

Left Arithmetic Shift- In this one position moves each bit to the left on by one. The empty least significant bit (LSB) is filled with zero and the most significant bit (MSB) is rejected.

Right Arithmetic Shift- In this one position moves each bit to the right on by one. The least significant bit (LSB) is rejected and the most significant bit (MSB) is filled with the value of the previous MSB.

Now, here given,

Initial value, R= 11000101

Now, applying operators,

Left Circular Shift = 10001011

Right Logical Shift = 01000101

Right Circular Shift = 10100010

Thus, final value of R = 01000100

Or in decimal we can write it as R = 68

Q.No.2.(a) What is addressing mode and why do we need addressing modes?

Ans. **ADDRESSING MODE-** The term addressing mode refers to the way in which the operand of an instruction is specified. The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually executed.

We need addressing modes because :-

1. To give programmers to facilities such as pointers, counters for loop controls, entering of data and program relocation.

2. To reduce the no. bits in the addressing field of the instruction.

(b) An instruction is stored in a location 450 with its address field at location 451. The address field has the value 800. A processor register R1 contains the no. 900. Evaluate the effective address if the addressing mode of the instruction is (i) Direct (ii) Immediate (iii) Relative (iv) Register indirect.

Ans. From the question,

Given, Instruction location = 450

Address field location = 451

Address field value = 800

Processor register R1 contains the number = 900

i. Now, If addressing mode of the instruction is direct, the effective address will be = 800.
ii. If addressing mode of the instruction is immediate, the effective address is memory address of 800 so it will be = 451.
iii. If the addressing mode of the instruction is relative, then effective address is sum of address field value and address of next instruction which is = 1252.
iv. If the addressing mode of the instruction is register indirect then the effective address is equal to the content of R1 register, which is = 900.

Q.No.3. Explain followings in detail: (i) Bootstrap Loader, (ii) Stored Program Concept, (iii) Difference between a branch instruction, a call sub-routine instruction and a program interrupt.

Ans. **(i) Bootstrap Loader–** The ROM portion of the main memory is needed for storing an initial program called a **Bootstrap Loader**. The Bootstrap Loader is a program whose function is to start the computer software operating when power is turned on. Since RAM is volatile, its contents are destroyed when power is turned off. The contents of ROM remain unchanged after power is turned off and on again. The start-up of a computer consists of turning the power on and starting the execution of an initial program. Thus, when power is turned on, the hardware of the computer sets the program counter to the first address of the bootstrap loader. The bootstrap program loads a portion of the operating system from disk to main memory and control is then transferred to the operating system, which prepares the computer for general use.

(ii) Stored Program Concept-

1. In the Stored Program Concept, both the instructions and the data (that the instruction operates on are stored in the computer memory itself. Before the introduction of this idea, instruction and data were considered two different entities and were thus stored separately.
2. Thus instructions like data can be read from the memory and written to the memory by the processor.
3. The processor then addresses the memory, reads the corresponding instructions, executes them and according to the executed instruction, processes (reads and writes) data as well.
4. Computers that store both instructions and data on the same memory are said to be based on the Von Neumann architecture. Modern desktop computers are still based on the same stored program concept.

(iii) Branch Instruction:- A branch is an instruction in a computer program that can cause a computer to begin executing a different instruction sequence and thus deviate from its default behaviour of executing instructions in order. Branch may also refer to the act of switching execution to a different instruction sequence as a result of executing a branch instruction. Branch instructions are used to implement control flow in program loops and conditions (i.e., executing a particular sequence of instructions only if certain conditions are satisfied).

Call Sub-Routine Instruction:- The instructions that transfer program control to a sub-routine are known by different names. The most common names used are called **sub-routine**, jump to a sub-routine, branch to a sub-routine, or branch and save the address. A call sub-routine instruction consists of an operation code together with an address that specifies the beginning of the sub-routine. The instruction is executed by performing two operations:

1. The address of the next instruction available in the program counter (the return address) is stored in a temporary location so the sub-routine knows where to return.
2. Control is transferred to the beginning of the sub-routine.

Program Interrupt:- The concept of program interrupt is used to handle a variety of problems that arise out of normal program sequences. **Program interrupt** refers to the transfer of program control from a currently running program to another service program as a result of an external or internal generated request. Control returns to the original program after the service program is executed.

Q.No.4. What is memory hierarchy and why do we need it? Explain.

Ans.

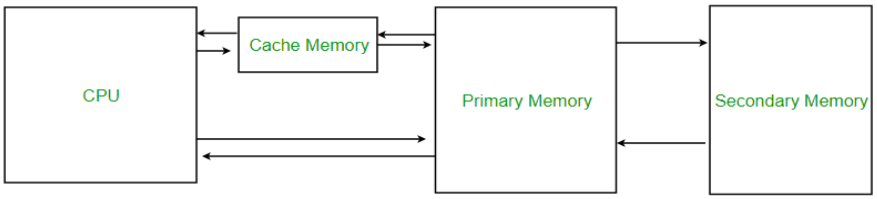
- CPU registers are at the top and fast cache memory is there between the CPU and main memory. The hard disk is used by the technique of virtual memory to expand the capacity of the main memory. In computer language, this kind of hierarchy is known as the **memory hierarchy** is applied to get larger memory space at a low cost.
- Not only that the fast memory has low storage capacity but it needs a power supply till the information needs to be stored and is costly.
- Memories with less cost have high access time. It is used to get a high access rate at the low cost of memory.
- As the storage capacity of the memory increases, the cost per bit for storing binary information decreases, and the access time of memory becomes longer. The auxiliary memory has a large storage capacity, is relatively inexpensive, but has low access speed compared to main memory.
- The cache memory is very small, relatively expensive, and has a very high access speed.
- Thus, as the memory access speed increases, so does its relative cost. The overall goal of using a memory hierarchy is to obtain the highest possible average access speed while minimizing the total cost of the entire memory system.

Q.No.5.(a) What is cache memory? Explain various cache mapping techniques with suitable examples.

Ans. Cache Memory :-

- It is a special very high-speed memory.
- It is used to speed up and synchronizing with high-speed CPU.
- It is costlier than main memory or disk memory but economical than CPU registers.
- It is an extremely fast memory type that acts as a buffer between RAM and the CPU.

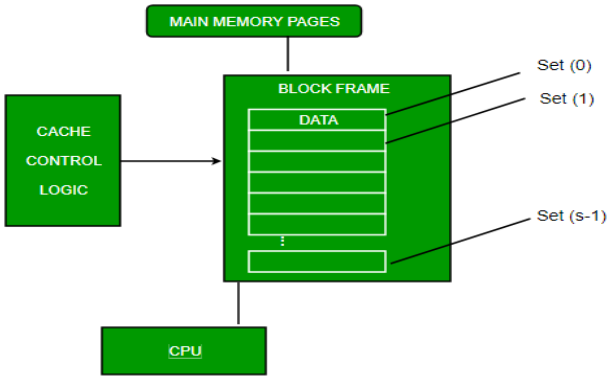
- It holds frequently requested data and instructions so that they are immediately available to the CPU when needed.
- It is used to reduce the average time to access data from the main memory.



Cache Mapping :-

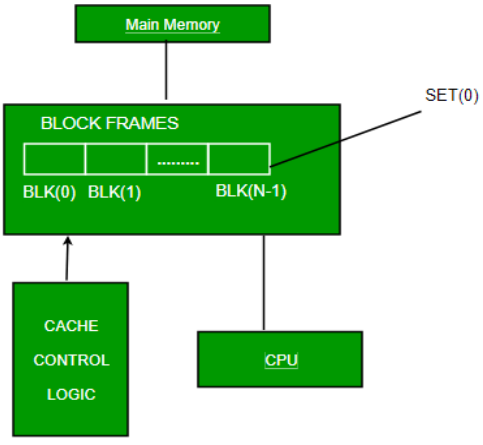
1. Direct Mapping - The simplest technique, known as **direct mapping**, maps each block of main memory into only one possible cache line.

In Direct mapping, assign each memory block to a specific line in the cache. If a line is previously taken up by a memory block when a new block needs to be loaded, the old block is trashed. An address space is split into two parts index field and a tag field. Direct mapping's performance is directly proportional to the Hit ratio.



2. Associative Mapping -

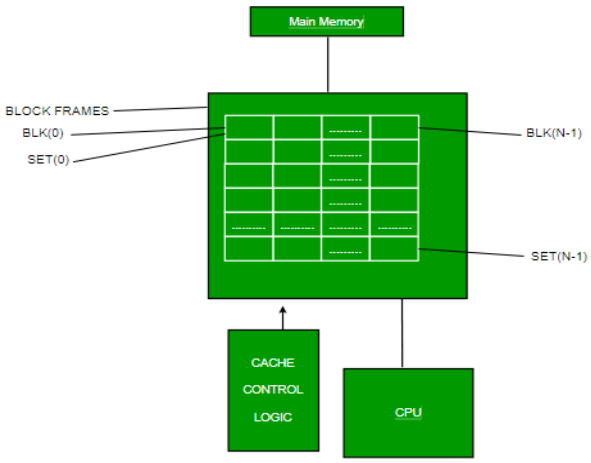
In this type of mapping, the associative memory is used to store content and addresses of the memory word.



Any block can go into any line of the cache. This means that the word id bits are used to identify which word in the block is needed, but the tag becomes all of the remaining bits. This enables the placement of any word at any place in the cache memory. It is considered to be the fastest and the most flexible mapping form.

3. Set-associative Mapping - It explains that instead of having exactly one line that a block can map to in the cache, we will group a few lines together creating a **set**. Then a block in memory can map to any one of the lines of a specific set.

This mapping allows that each word that is present in the cache can have two or more words in the main memory for the same index address. Set associative cache mapping combines the best of direct and associative cache mapping techniques.



(b) A computer system has physical memory of size 4Giga words. Direct-mapped cache memory can store 256 blocks. A block can store 16 words. You have to find the line no. in cache memory where each of the following addresses can be mapped: (i) 2D345C12 (ii) F45F40F5 (iii) 12565348

Ans. main memory, words= 4G= 2^32

cache, words= 2^8 blocks

block size= 2^4 words

(given)

blocks in main memory= 2^32 words / 2^4 words = 2^28 blocks

blocks in cache memory= 2^8 blocks

address format -> block no. + word-offset

-> tag + line no. + word-offset

no. of words in each block = 2^4

no. of lines in cache = 2^8

no. of words in main memory= 2^32

so address format= 20 bits tag + 8 bits line no. + 4 bits word-offset

(i) address= 2D345C12

= 2D345 | C1 | 2

20bits 8bits 4bits

line number = C1 = 12*16+1 = 181

(ii) address= F45F40F5

= F45F4 | 0F | 5

20bits 8bits 4bits

line number = 0F = 0*16+15 = 15

(iii) address= 12565348

= 12565 | 34 | 8

20bits 8bits 4bits

line number = 34 = 3*16+4 = 52

Final answer :

(i) address = 2D345C12 maps to line no. 181

(ii) address = F45F40F5 maps to line no. 15

(iii) address = 12565348 maps to line no. 52

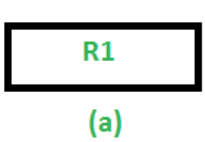
Q.No.6. Write short notes on: (i) Register Transfer Language, (ii) Serial Communication, (iii) Strobe Control, (iv) Handshaking.

(i) Register Transfer Language :-

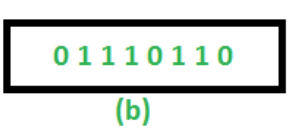
The **Register Transfer Language** is the symbolic representation of notations used to specify the sequence of micro-operations. It is a kind of intermediate representation (IR) that is very close to assembly language, such as that which is used in a compiler.

There are various methods of RTL :

1. General way of representing a register is by the name of the register enclosed in rectangular box as shown in (a).
2. Register is numbered in a sequence of 0 to (n-1) as shown in (b).
3. The numbering of bits in a register can be marked on the top of the box as shown in (c).
4. A 16-bit register PC is divided into 2 parts- Bits (0 to 7) are assigned with lower byte of 16-bit address and bits (8 to 15) are assigned with higher bytes of 16-bit address as shown in (d).

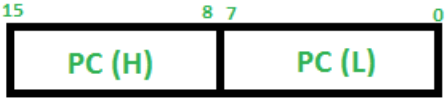


(a)



(b)

(c)



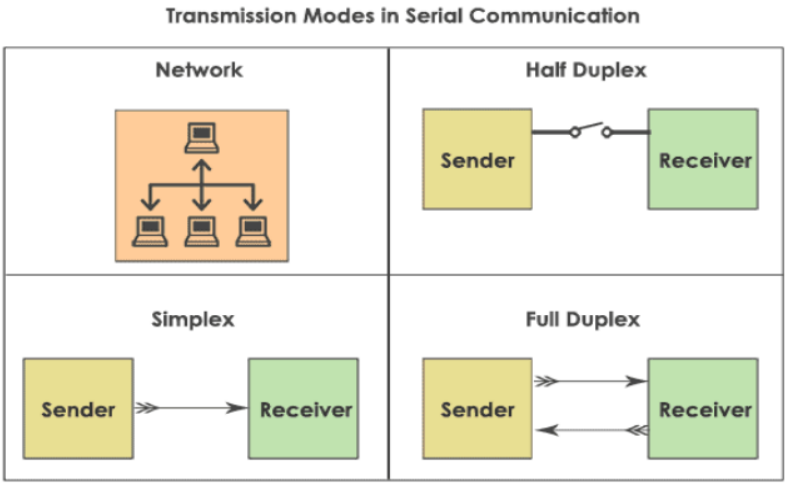
(d)

Basic symbols of RTL:

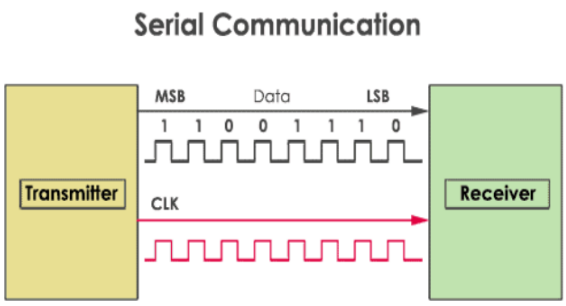
Symbol	Description	Example
Letters and Numbers	Denotes a Register	MAR, R1, R2
( )	Denotes a part of register	R1(8-bit) R1(0-7)
<-	Denotes a transfer of information	R2 <- R1
:	Denotes conditional operations	P : P : R2 <- R1 if P=1

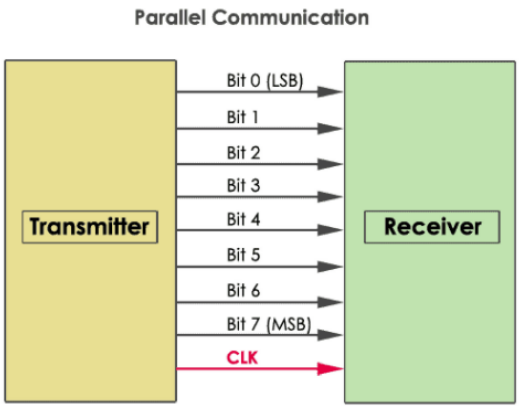
(ii) Serial Communication :-

Serial communication is the way of exchanging data using different methods in the form of serial digital binary.
Serial communication is the most widely used approach to transfer information between data processing equipment and peripherals. In other words, we can say Binary 'one' represents a logic HIGH or 5 Volts, and 'zero' represents a logic LOW or 0 Volt.
It can take many forms depending on the type of **transmission mode** and **data transfer**.
The **transmission modes** are classified as Simplex, Half Duplex, and Full Duplex. There will be a source (also known as a **sender**) and destination (also called a **receiver**) for each transmission mode.



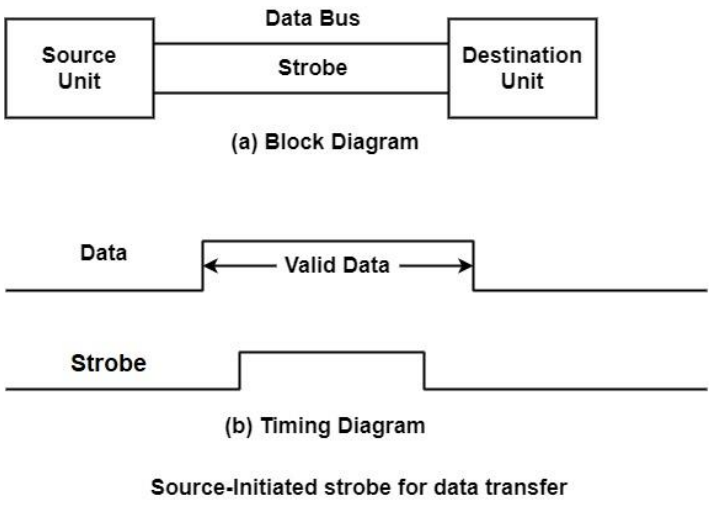
Data transfer can happen in two ways. They are **serial communication** and **parallel communication.**





(iii) Strobe Control :-

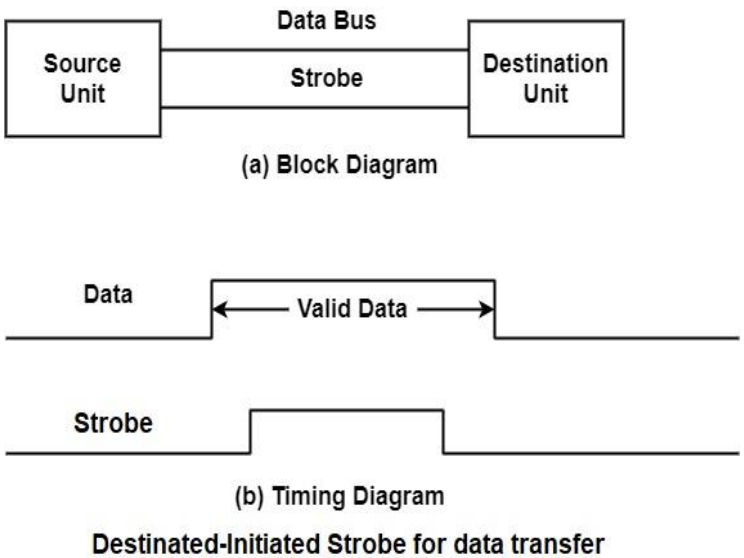
The **strobe control** technique of asynchronous data transfer operates a single control line to time each transfer. The strobe can be activated by either the source or the destination unit.



Source-Initiated strobe for data transfer

The data bus gives the binary data from the source unit to the destination unit. Generally, the bus has multiple lines to transfer a unified byte or word. The destination unit helps the lowering edge of the strobe pulse to send the contents of the data bus into one of its internal registers.

The source deletes the data from the bus for a short period after it disables its strobe pulse.

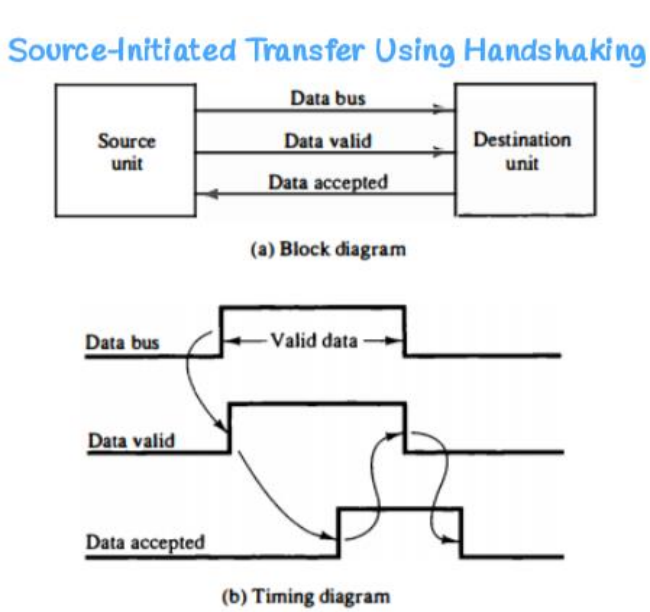


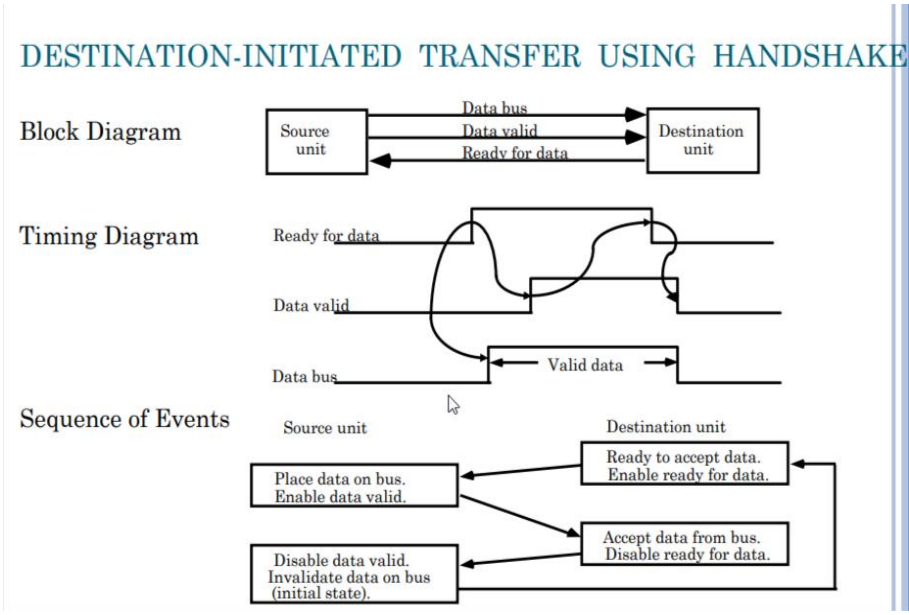
Destinated-Initiated Strobe for data transfer

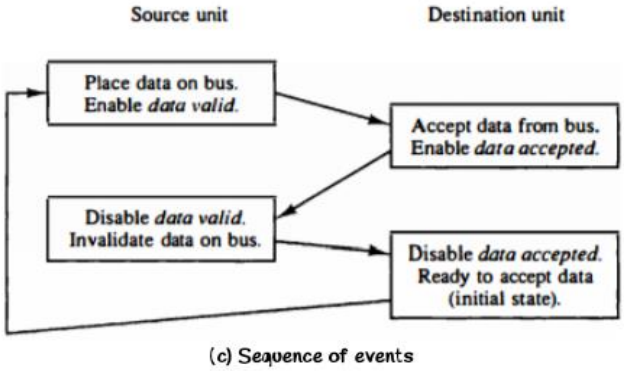
In this method, the destination unit activates the strobe pulse, advising the source to support the data. The source unit counter by storing the requested binary data on the data bus. The data should be true and remain in the bus long sufficient for the destination unit to accept it. The falling edge of the strobe pulse can be used to produce a destination register. The destination unit then disables the strobe.

(iv) Handshaking :-

Handshaking is an I/O control approach to synchronize I/O devices with the microprocessor. As several I/O devices accept or release data at a much lower cost than the microprocessor, this technique is used to control the microprocessor to operate with an I/O device at the I/O devices data transfer rate.







Handshaking solves the issue of **Strobe Control** by introducing a second control signal that supports a response to the unit that initiates the transfer. The sequence of control during the transfer is based on the unit that initiates the transfer.

The sequence of events in both cases would be equal if it can consider the ready-for-data signal as the complement of data accepted.

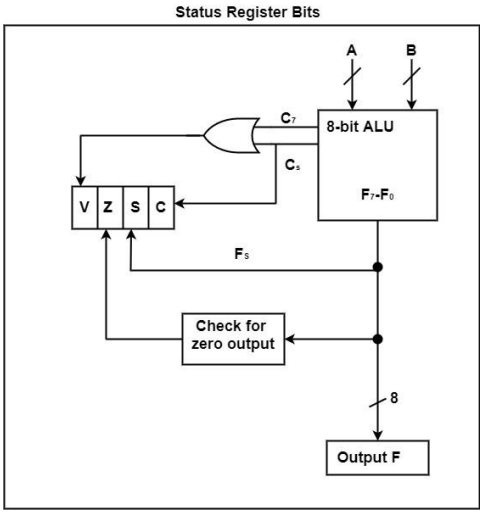
The only difference between the source-initiated and the destination-initiated transfer is in their choice of the initial state.

Q.No.7.(a) What are the various status bit conditions stored in a status register. How do they help in conditional branch instructions? Explain with suitable examples.

Ans. After ALU operation the status of CPU change. To specify the status of CPU after each operation we use the term **Status Bit Condition.** It is also known as **Flag Bits or Condition Codes**.

- The collection of all status bits condition is called **Program Status Word (PSW).**
- The **PSW** is stored in a special hardware register, that's called **Status Register.**

The status register comprises the status bits. The bits of the status register are modified according to the operations performed in the ALU**.**
The figure displays a block diagram of an 8-bit ALU with a 4-bit status register :



Common Status Flag :-

1. Common Flag (c) :

Set to 1, if carry is generated after ALU operation else reset to zero.

c=1, if carry generates

c=0, if carry doesn't generate

2. Sign Flag (s) :

s=1, if MSB is 1, -ve

s=0, if MSB is 0, +ve

3. Zero Flag (z) :

z=1, when all the bits are zero after ALU operation

z=0, otherwise

4. Overflow Flag (o) :

v=1, **xor** of two left most bits are 1

v=0, **xor** of two left most bits are 0

Eg. let A= 10110110

B= 11001011

ALU operation: OR (say)

A+B= 10110110 + 11001011 = 110000001

Here, **c=1, s=1, z=0, v=1**

Conditional Branch Instruction :

- It's a type of branch instructions.
- Instructions which disturb the normal flow of program.
- It has an operand which is the address of next instruction to execute and out of sequence.
- We need to store the current status of CPU . Then we use **PSW** of status bit. And when we need to access it, we can do from status register.
- If certain condition is true :
  - It changes the content of PC.
  - Conditions are dictated by conditional flags.
  - Bits of status register.
    NZ(Z=0), Z(Z=1), NC(C=0), C(C=1)

Conditional Instruction :

Mnemonic	Condition	Tested Condition
BZ	Branch if zero	Z=1
BNZ	Branch if not zero	Z=0
BC	Branch if carry	C=1
BNC	Branch if no carry	C=0
BM	Branch if minus	S=1
BV	Branch if overflow	V=1
BNV	Branch if no overflow	V=0

Unsigned compare conditions (A –B) :

BHI	Branch if higher	A > B
BHE	Branch if higher or equal	A ≥ B

Various condition can be generated using other flags.

(b) The memory unit of computer has 256K words of 64 bits each. The computer has an instruction format with four fields: an operation code field, a mode field specify one of 13 addressing mode, a register address field to specify one of 80 processor register, and a memory address. Specify the instruction format and the no. of bits in each field if the instruction is in one memory word.

Ans. It's an addressing instruction.

Instruction Size= 64 bits

Memory address bits -> 256K words = 18 bits

Register address bits -> 80 registers = 7 bits

Mode bit -> 13 addressing modes = 4 bits

Opcode size = 64 - (18+7+4) = 35

Q.8. (a) Why I/O interface is required? Give any three reasons.

Ans. Devices connected to processor externally are known as **Peripherals.**

CPU can't access directly I/O devices. So, CPU will be connected with I/O devices using intermediating which is called **Interface**.

There are following reasons of requirement of interface :-

a) Whichever CPU generate signal. It is not directly understand by I/O devices and vice versa. Hence, there is need to convert their language to understand each other.
b) The data transfer rate of peripherals is usually slow. So, synchronisation (by interface) is required.
c) The operating modes of peripherals are different from each other and each must be controlled. So, a peripheral doesn't disturb the operation of other peripherals.

(b) Explain various modes to handle data transfer between computer and I/O devices.

Ans. The binary information that is received from an external device is usually stored in the memory unit. The information that is transferred from the CPU to the external device is originated from the memory unit.

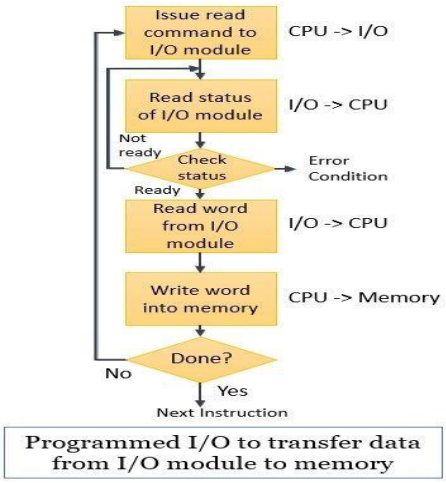
Data transfer between computer and I/O devices be handle in mainly 3 modes :-

1. Programmed I/O
2. Interrupt-initiated I/O
3. Direct Memory Access (DMA)

1. Programmed I/O :-

- There is no any provision through which I can inform to CPU about data transfer.

- CPU runs program periodically and check the status of each devices one by one.
- If any device has its status bit, then CPU perform data transfer for it.
- It's more time consuming mode.



Programmed I/O to transfer data from I/O module to memory

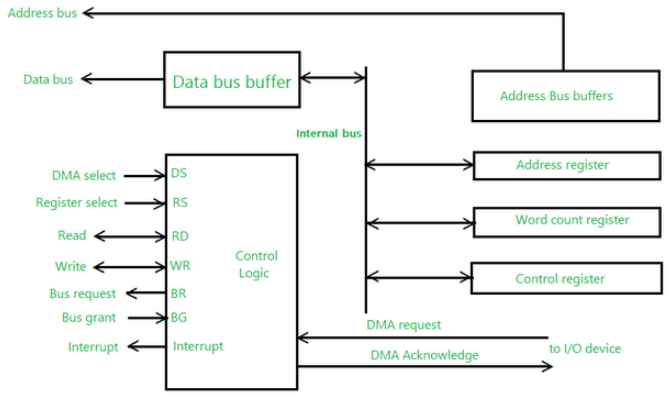
CPU makes a request and then CPU stages in program loop (pooling) continue checking status until I/O devices indicate that it's ready for data transfer.

2. Interrupt-initiated I/O :-

- I/O devices has a provision (interrupt signal by interference) to inform to CPU about communication.
- When CPU received interrupt :
  - ➢ It completes execution of current instruction.
  - ➢ Saves and status of current prices process on to the stack.
  - ➢ Resume the previous process by taking out the value from stack.
- Using interrupt mode the waiting time period of CPU is ideally removed.

3. Direct Memory Access (DMA) :-

- It enables data transfer between I/O and memory without CPU intervention.
- Need a hardware : **DMAC**
- In **DMA,** CPU grands to I/O interface of authority to read from or write to memory without its involvement.



Modes of DMA :

- Burst Mode
- Cycle Stealing
- Interleaving Mode :

Percentage of time that CPU is blocked due to DMA= 0.