
CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A++ GRADE BY NAAC

Internal Assessment Test 2 – DEC 2024

Sub:	**Digital Design and Computer Organization**				Sub Code:	**BCS302**	Branch:	**CS(DS)/AInDS**
Date:	**13/12/2024**	Duration:	**90 minutes**	Max Marks:	**50**	Sem	**III**	**OBE**

Answer any FIVE Questions			**MARKS**	**CO**	**RBT**
1		What are Addressing Modes and explain all.	[10]	3	L2
2	a	Explain Basic Instruction Types.	[5]	3	L2
	b	Analyze Big –endian and little –endian methods of byte addressing with Examples	[5]	3	L3
3	a	Give a detailed description of the execution of complete instruction.	[5]	5	L2
	b	Explain the centralised and distributed bus arbitration schemes with a neat diagram.	[5]	4	
4	a	Demonstrate the DMA and its implementation and show how the data is transferred between memory and I/O devices using the DMA controller.	[5]	4	L3
	b	Discuss Cache memory and Mapping functions. Explain the direct, associative and set-associative mapping with a labelled diagram.	[5]		
5	a	What is an interrupt? What are interrupt service routines and what are vectored interrupts? Explain with an example.	[5]	4	L2
	b	Discuss Register transfers, performing an Arithmetic or logic operation, Fetching a word from memory and storing a word in memory.	[5]	5	
6	a	What is instruction pipelining?	[4]	5	L3
	b	What is the role of cache memory? Discuss pipeline performance.	[6]	5	

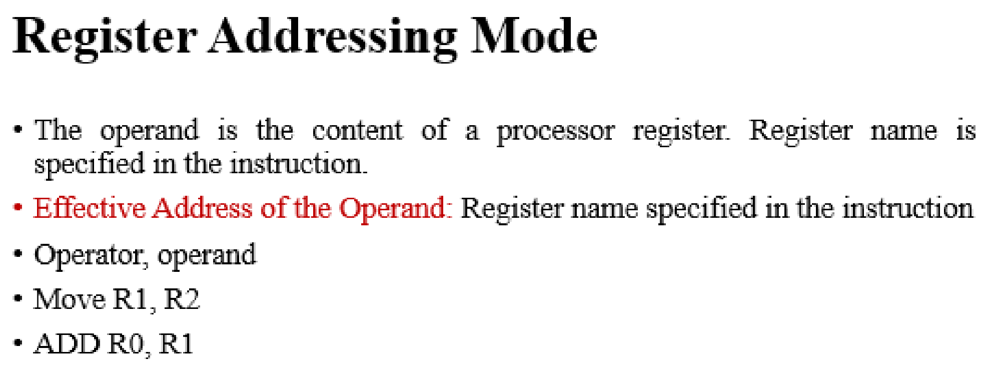
IAT 2 Answer Sheet

Q. 1 What is an Addressing Modes (2) and explain all (8).

Ans. The term addressing modes refers to how the operand of an instruction is specified. Information contained in the instruction code is the value of the operand or the address of the operand. Following are the main addressing modes that are used on various platforms and architectures.

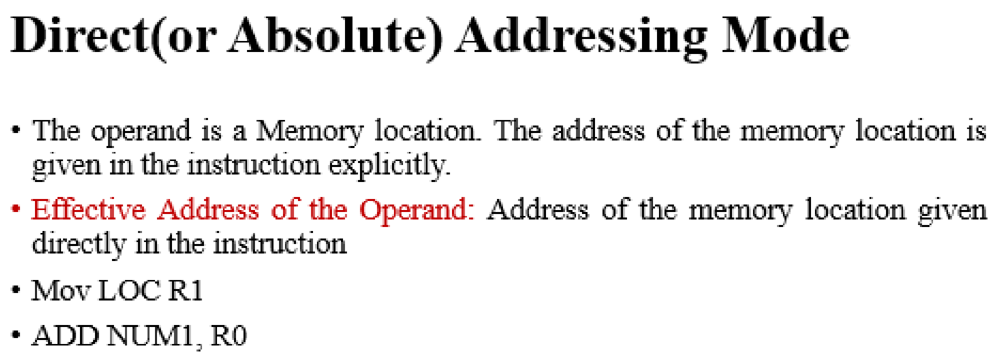
1. Register Addressing Mode
2. Immediate Addressing Mode
3. Direct (or Absolute) Addressing Mode

4. Indirect Addressing Mode
5. Index Addressing Mode
6. Relative Addressing Mode
7. Auto increment Addressing Mode
8. Auto decrement Addressing Mode.



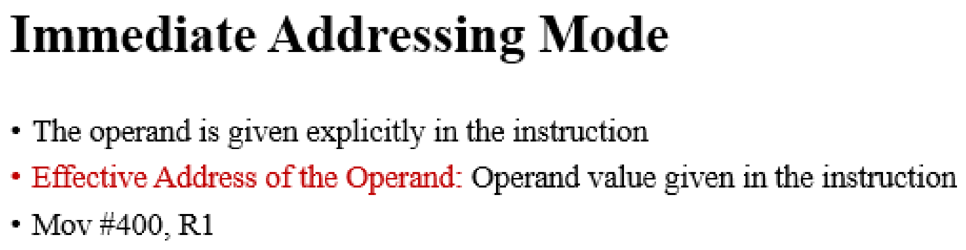
Register Addressing Mode

- The operand is the content of a processor register. Register name is specified in the instruction.
- Effective Address of the Operand: Register name specified in the instruction
- Operator, operand
- Move R1, R2
- ADD R0, R1



Direct(or Absolute) Addressing Mode

- The operand is a Memory location. The address of the memory location is given in the instruction explicitly.
- Effective Address of the Operand: Address of the memory location given directly in the instruction
- Mov LOC R1
- ADD NUM1, R0



Immediate Addressing Mode

- The operand is given explicitly in the instruction
- Effective Address of the Operand: Operand value given in the instruction
- Mov #400, R1

Indirect Addressing Mode

- Here neither the operands nor the addresses are given explicitly. The instruction provides the information from which the address of the operand is determined i.e., the instruction provides effective address of the operand using register or memory location. The indirection is denoted by () sign around register or memory.
- Effective Address of the Operand: (Ri) or (LOCA) is the contents of a register or the memory location whose address appears in the instruction

Index Addressing Mode

- The effective address of the operand is generated by adding a constant value to the contents of a register specified in the instruction. The register in this case is called as Index register.
- The operation is indicated as X(Ri).
- Effective Address of the Operand: X+Ri where X is a constant value (signed integer) and Ri is the index register.

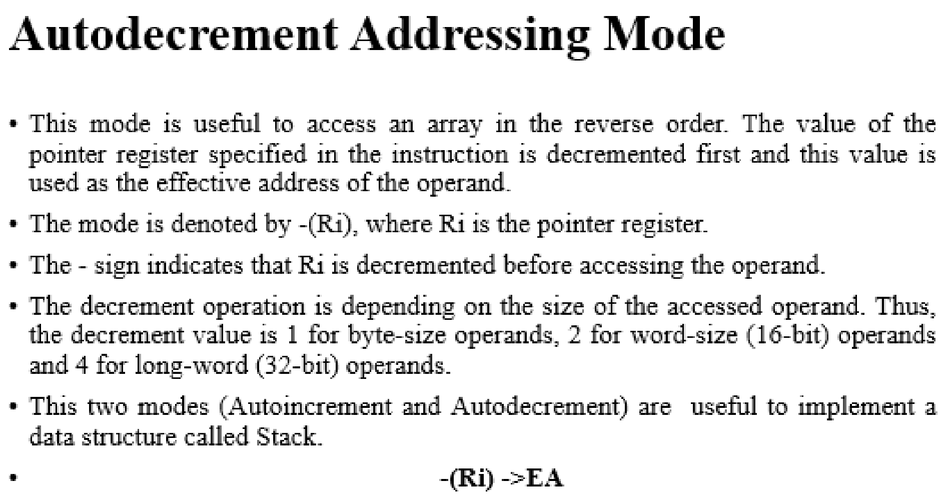
Relative Addressing Mode

- In this mode the content of the program counter is added to the address part of the instruction to obtain the effective address.
- Effective Address : X+PC where X is a constant value (signed integer) and PC is the contents of the program counter.
- Program counter is responsible to carry out the contents of the operands

Autoincrement Addressing Mode

- This is indirect mode with a modification. The effective address of the operand is the contents of a pointer register specified in the instruction. After accessing the operand, the contents of this pointer register is incremented automatically to point to the next entity.
- The mode is denoted by (Ri)+, where Ri is the pointer register.
- The + sign indicates that Ri is incremented after the operation.
- The increment operation is depending on the size of the accessed operand. Thus, the increment value is 1 for byte-size operands, 2 for word-size (16-bit) operands and 4 for long-word (32-bit) operands.
- This mode is useful when operands are stored consecutively in memory i.e., for array manipulation
- **contents of the register**

(Ri)+ ->EA



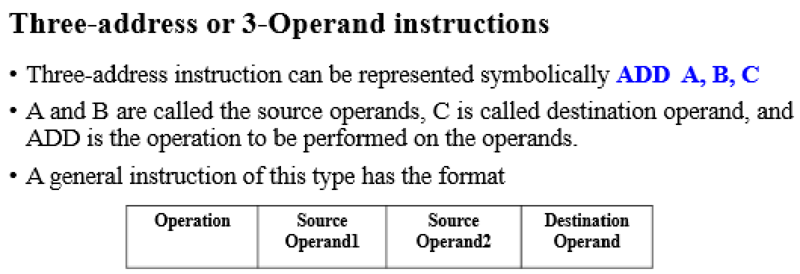
Autodecrement Addressing Mode

- This mode is useful to access an array in the reverse order. The value of the pointer register specified in the instruction is decremented first and this value is used as the effective address of the operand.
- The mode is denoted by -(Ri), where Ri is the pointer register.
- The - sign indicates that Ri is decremented before accessing the operand.
- The decrement operation is depending on the size of the accessed operand. Thus, the decrement value is 1 for byte-size operands, 2 for word-size (16-bit) operands and 4 for long-word (32-bit) operands.
- This two modes (Autoincrement and Autodecrement) are useful to implement a data structure called Stack.
- **-(Ri) ->EA**

Q.2 a) Explain Basic Instruction Types (5).

Ans. Instruction Set Categories based on the Operands explicitly specified in the instruction.

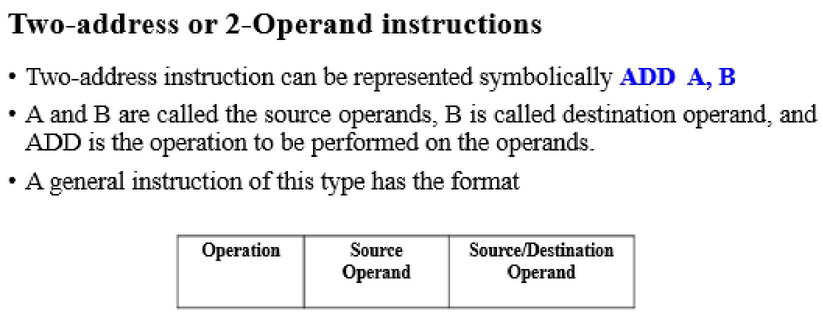
1. Three-address or 3-Operand instructions
2. Two-address or 2-Operand instructions
3. One-address or 1-Operand instructions
4. Zero-address or 0-Operand instructions



Three-address or 3-Operand instructions

- Three-address instruction can be represented symbolically **ADD A, B, C**
- A and B are called the source operands, C is called destination operand, and ADD is the operation to be performed on the operands.
- A general instruction of this type has the format

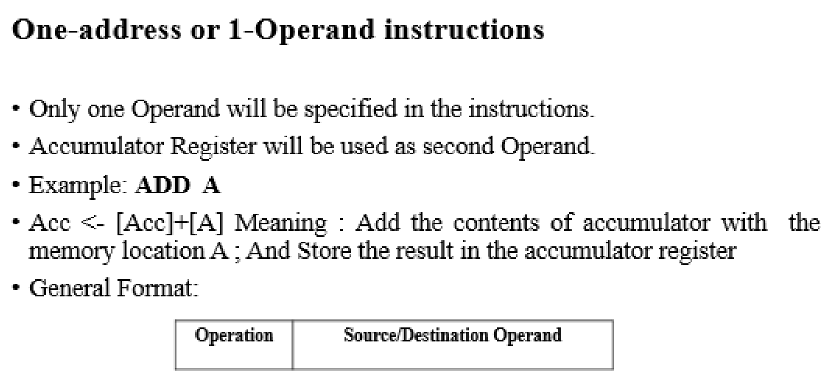
Operation	Source Operand1	Source Operand2	Destination Operand



Two-address or 2-Operand instructions

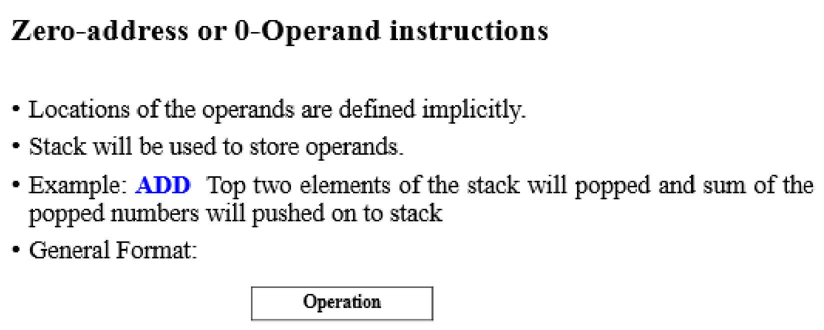
- Two-address instruction can be represented symbolically **ADD A, B**
- A and B are called the source operands, B is called destination operand, and ADD is the operation to be performed on the operands.
- A general instruction of this type has the format

Operation	Source Operand	Source/Destination Operand


One-address or 1-Operand instructions

- Only one Operand will be specified in the instructions.
- Accumulator Register will be used as second Operand.
- Example: **ADD A**
- Acc <- [Acc]+[A] Meaning : Add the contents of accumulator with the memory location A ; And Store the result in the accumulator register
- General Format:

Operation	Source/Destination Operand


Zero-address or 0-Operand instructions

- Locations of the operands are defined implicitly.
- Stack will be used to store operands.
- Example: **ADD** Top two elements of the stack will popped and sum of the popped numbers will pushed on to stack
- General Format:

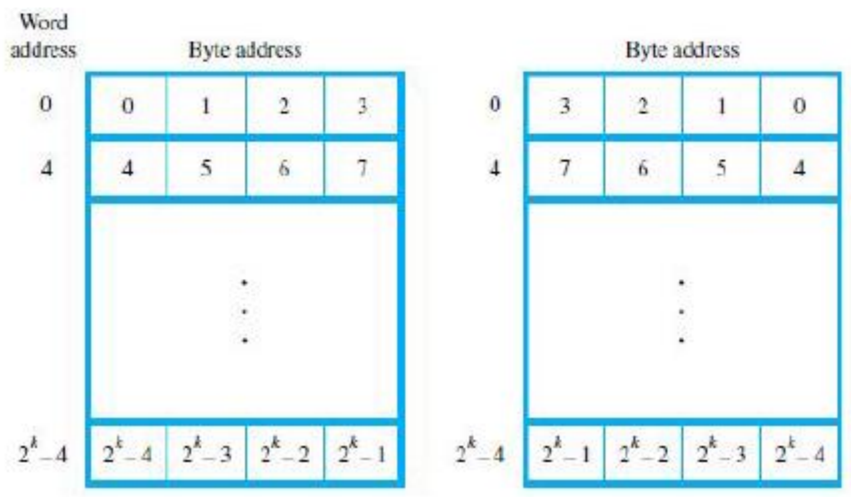
Operation

b) Analyze Big –endian (2) and little –endian (2) methods of byte addressing with Example (1).

Ans. Big-Endian: lower byte addresses are used for the most significant bytes of the word.

Little-Endian: opposite ordering. Lower byte addresses are used for the less significant bytes of the word.

In both cases, byte-addresses 0, 4, 8.....are taken as the addresses of successive words in the memory.



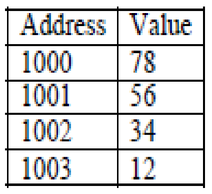
(a) Big-endian assignment (b) Little-endian assignment

Example: Consider a 32-bit integer (in hex): 0 **x** 12345678 which consists of 4 bytes: 12, 34, 56, and78.

• Hence this integer will occupy 4 bytes in memory.

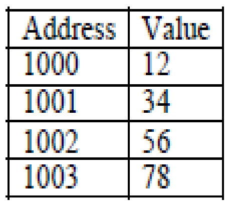
• Assume, we store it at memory address starting 1000.

On little-endian, memory will look like



Address	Value
1000	78
1001	56
1002	34
1003	12

On big-endian, memory will look like



Address	Value
1000	12
1001	34
1002	56
1003	78

Q. 3 a) Give a detailed description of the execution of a complete instruction.

Ans. Add (R3), R1

- Fetch the instruction
- Fetch the first operand (the contents of the memory location pointed to by R3)
- Perform the addition
- Load the result into R1
- Add (R3), R1

Step	Action
1	PC_{out} , MAR_{in} , Read, Select4, Add, Z_{in}
2	Z_{out} , PC_{in} , Y_{in} , WMF C
3	MDR_{out} , IR_{in}
4	$R3_{out}$, MAR_{in} , Read
5	$R1_{out}$, Y_{in} , WMF C
6	MDR_{out} , SelectY, Add, Z_{in}
7	Z_{out} , $R1_{in}$, End

Figure 7.6. Control sequencefor execution of the instruction Add (R3),R1.

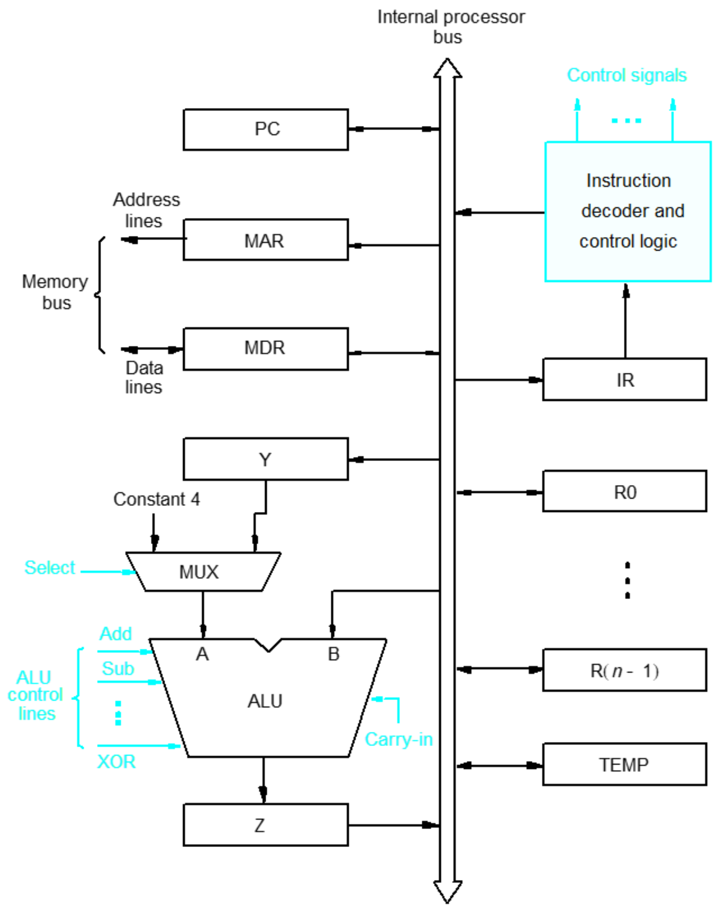


Figure 7.1. Single-bus organization of the datapath inside a proce

b) Explain the centralised and distributed bus arbitration schemes with a neat diagram.

Ans. Bus Arbitration- The device that is allowed to initiate data-transfers on bus at any given time is called bus master.

- There can be only one bus-master at any given time.

- Bus Arbitration is the process by which next device to become the bus-master is selected & bus-mastership is transferred to that device.
- The two approaches are:

1) **Centralized Arbitration:** A single bus-arbiter performs the required arbitration.

- A single bus-arbiter performs the required arbitration.
- Normally, processor is the bus-master.
- Processor may grant bus-mastership to one of the DMA controllers.
- A DMA controller indicates that it needs to become bus-master by activating BR line.
- The signal on the BR line is the logical OR of bus-requests from all devices connected to it.
- Then, processor activates BG1 signal indicating to DMA controllers to use bus when it becomes free.

2) **Distributed Arbitration**: All devices participate in selection of next bus-master.

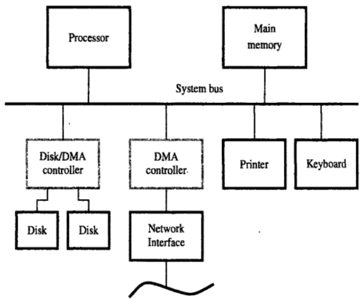
- All device participate in the selection of next bus-master
- Each device on bus is assigned a 4-bit identification number (ID).
- When 1 or more devices request bus, they → assert Start-Arbitration signal & place their 4-bit ID numbers on four open-collector lines 0 BRA through 3 BRA .
- A winner is selected as a result of interaction among signals transmitted over these lines.
- Net-outcome is that the code on 4 lines represents request that has the highest ID number.
- Advantage: This approach offers higher reliability since operation of bus is not dependent on any single device.

Q. 4 a) Demonstrate the DMA and its implementation and show how the data is transferred between memory and I/O devices using the DMA controller.

Ans. The transfer of a block of data directly b/w an external device & main-memory w/o continuous involvement by processor is called DMA.

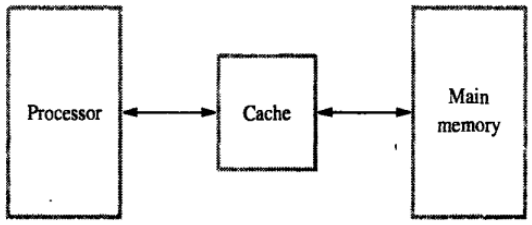
- DMA controller→ is a control circuit that performs DMA transfers. → is a part of the I/O device interface. → performs the functions that would normally be carried out by processor.
- While a DMA transfer is taking place, the processor can be used to execute another program.

- DMA interface has three registers: 1) First register is used for storing starting-address. 2) Second register is used for storing word-count. 3) Third register contains status- & control-flags.
- The R/W bit determines direction of transfer. If R/W=1, controller performs a read-operation (i.e. it transfers data from memory to I/O), Otherwise, controller performs a write-operation (i.e. it transfers data from I/O to memory).
- If Done=1, the controller has completed transferring a block of data and is ready to receive another command. (IE - Interrupt Enable).
- If IE=1, controller raises an interrupt after it has completed transferring a block of data.
- If IRQ=1, controller requests an interrupt.
- Requests by DMA devices for using the bus are always given higher priority than processor requests.

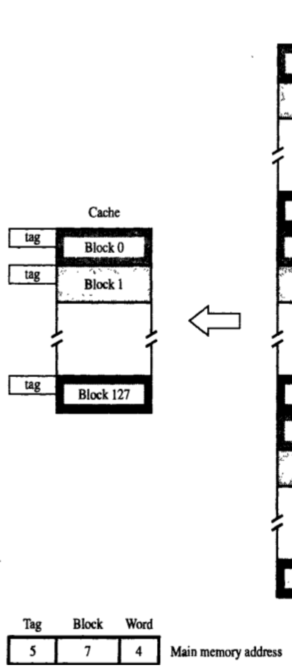


b) Discuss Cache memory and Mapping functions. Explain the direct, associative and set-associative mapping with a labelled diagram.

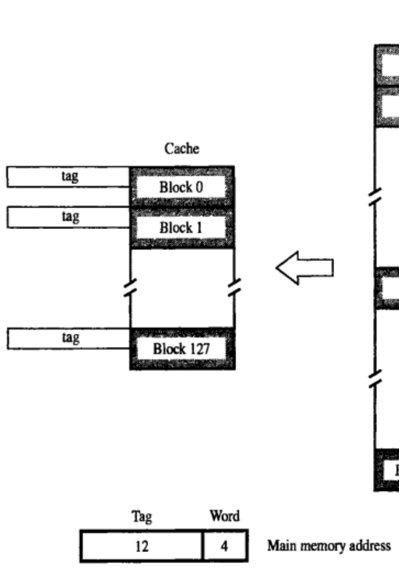
Ans. Cache Memory- The speed of the main memory is very low in comparison with the speed of modern processors. For good performance, the processor cannot spend much of its time waiting to access instructions and data in main memory. An efficient solution is to use a fast cache memory which makes the memory appear to the processor to be faster than it really is.



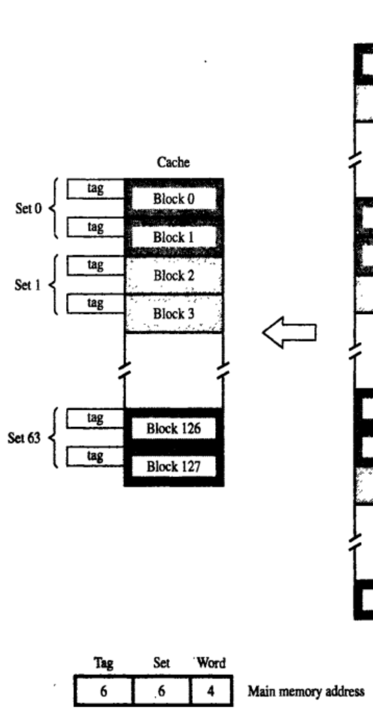
- Direct Mapping – The simplest way to determine cache locations in which to store memory blocks is the direct-mapping technique.



- Associative Mapping – It gives complete freedom in choosing the cache location in which to place the memory block.



- Set-Associative Mapping – It is the combination of direct and associative mapping techniques. Blocks of the cache are grouped into sets, and the mapping allows a block of the main memory to reside in any block of a specific set.



Q. 5 a) What is an interrupt? What are interrupt service routines and what are vectored interrupts? Explain with an example.

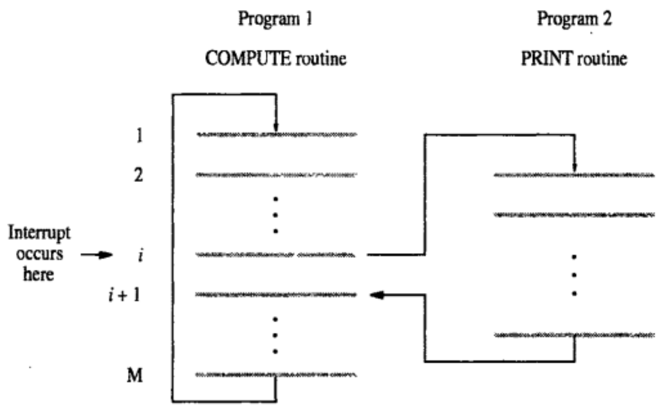
Ans. • I/O device initiates the action instead of the processor. This is done by sending a special hardware signal to the processor called as interrupt (INTR), on the interrupt-request line.

• The processor can be performing its own task without the need to continuously check the I/O device.

• When device gets ready, it will "alert" the processor by sending an interrupt-signal.

• The routine executed in response to an interrupt-request is called ISR (Interrupt Service Routine).

Once the interrupt-request signal comes from the device, the processor has to inform the device that its request has been recognized and will be serviced soon. This is indicated by a special control signal on the bus called interrupt-acknowledge (INTA).



Vectored Interrupt-

- A device requesting an interrupt identifies itself by sending a special-code to processor over bus.
- Then, the processor starts executing the ISR.
- The special-code indicates starting-address of ISR.
- The special-code length ranges from 4 to 8 bits.
- The location pointed to by the interrupting-device is used to store the staring address to ISR.
- The staring address to ISR is called the interrupt vector.
- Processor loads interrupt-vector into PC & executes appropriate ISR.
- When processor is ready to receive interrupt-vector code, it activates INTA line.

b) Discuss Register transfers, performing an Arithmetic or logic operation, Fetching a word from memory and storing a word in memory.

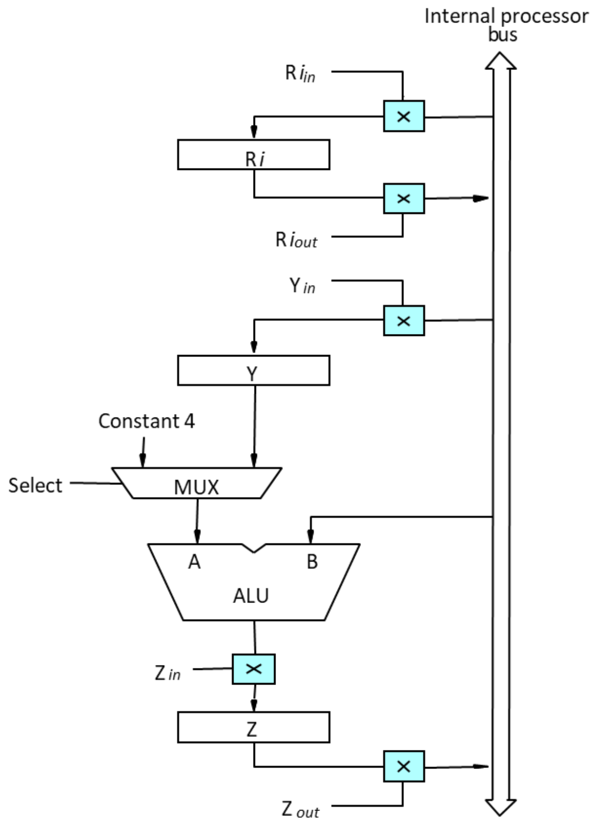


Figure 7.2. Input and output gating for the registers in Figure 7.1.

Ans.

Transfer the contents of R1 to R4.

1. Enable output of register R1 by setting R1out=1. This places the contents of R1 on the processor bus.
2. Enable input of register R4 by setting R4in=1. This loads the data from the processor bus into register R4.

The ALU is a combinational circuit that has no internal storage. ALU gets the two operands from MUX and bus. The result is temporarily stored in register Z. What is the sequence of operations to add the contents of register R1 to those of R2 and store the result in R3?

R1out, Yin

R2out, SelectY, Add, Zin

Zout, R3in

Register Transfer:- All operations and data transfers are controlled by the processor clock.

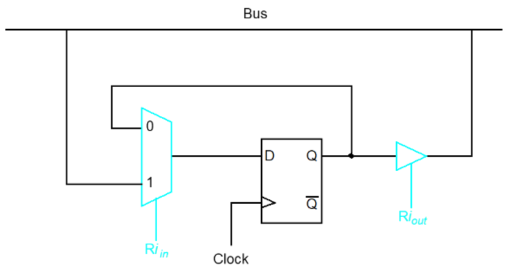


Figure 7.3. Input and output gating for one register bit.

Address into MAR; issue Read operation; data into MDR

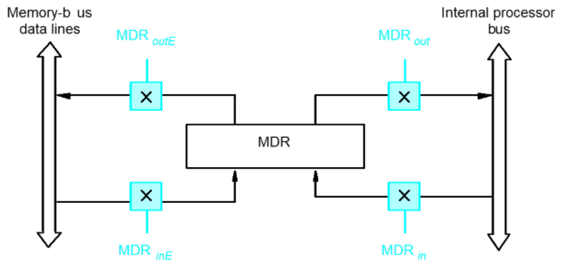


Figure 7.4. Connection and control signals for re gister MDR.

Address is loaded into MAR

Data to be written loaded into MDR.

Write command is issued.

Example: Move R2,(R1)

$R1_{out}$,MAR_{in}

$R2_{out}$,MDR_{in},Write

MDR_{outE}, WMFC

Q. 6 a) What is instruction pipelining?

Ans. An arithmetic pipeline divides an arithmetic problem into various sub problems for execution in various pipeline segments. It is used for floating point operations, multiplication and various other computations.

Floating point addition using arithmetic pipeline :

The following sub operations are performed in this case:

1. Compare the exponents.
2. Align the mantissas.
3. Add or subtract the mantissas.
4. Normalise the result

First of all the two exponents are compared and the larger of two exponents is chosen as the result exponent. The difference in the exponents then decides how many times we must shift the smaller exponent to the right. Then after shifting of exponent, both the mantissas get aligned. Finally the addition of both numbers take place followed by normalisation of the result in the last segment.

b) What is the role of cache memory? Discuss pipeline performance.

Ans. Cache memory is a very fast type of memory used by the computer to store frequently accessed data and instructions. Basically, it stores important data that the CPU needs often, so it can access it much faster than from the regular memory. This helps the computer run faster and more smoothly.

The CPU breaks down the process of executing an instruction (like adding two numbers or fetching data) into several stages, such as:

1. **Fetch** the instruction.
2. **Decode** the instruction (figure out what to do).
3. **Execute** the instruction (perform the operation).
4. **Store** the result (save the output)

Reducing Memory Latency: By providing fast access to frequently required instructions and data.

Improving processor's performance: helps the CPU get what it needs quickly, allowing pipelining to work more efficiently without unnecessary delays.

Think of the CPU as a worker on an assembly line. The worker needs parts (instructions and data) to complete tasks. If the worker has to go far away to get the parts (like accessing slow main memory), the whole line slows down.

Cache memory acts like a small, quick storage area right next to the worker. When the worker (the CPU) needs parts (instructions or data), the cache provides them **immediately** without needing to go all the way to the main storage. This allows the worker to keep working without interruptions.