Computer Organization and Architecture

This Chapter "Computer Organization and Architecture" **is taken from our:**

ISBN : 9789386629029

2

Page - 47 - 118

COMPUTER ORGANIZATION AND ARCHITECTURE

CONTENTS

MACHINE INSTRUCTIONS

A Computer needs to perform various different kind of tasks. To perform these diverse tasks, a computer needs various instructions that interact with low level hardware. For example to set some color on a part of display screen a computer needs an instruction. To read a single character from a keyboard the computer need other instruction. Similarly to perform arithmetic and logical operations like addition, subtraction, complement etc it needs a set of instructions. Although different computer have different instruction sets, yet the basic instruction are more or less same, the difference lies between the binary encoding of the instructions on different machines. For ex. Bit sequence "10011001" may mean ADDITION on a machine while "Complement operation" on the other. In any case there is a set of basic operations that most computers include in their instruction set. We can classify the instructions of a computer in following types :

1. **Data transfer instructions** : copy data from one to another.
2. **Data manipulation instructions** : Use existing bit patterns to compute a new bit patterns.
3. **Program control instructions** : Direct the execution of the program.

Instructions that perform arithmetic, logic, and shift operations are called Data Manipulation Instructions. Instructions like Conditional Branch, Jump instruction, subroutine call are included in Program control instructions that provide decision-making capabilities and change the path taken by the program when executed in the computer.

1. Data Transfer Instructions :

Data transfer instructions are used to transfer of data from one address to another without changing the contents. Mostly the transfer of data occurs between two registers of the processor, between main memory (often called RAM) and the registers of processor and between processor registers and I/O Devices. Data can be transferred between registers or between register and the memory. Following data transfer instructions are commonly used:

Data Transfer Instructions

Name	Mnemonic	Operation
Load	LD	Transfers data from a memory location to a processor register
Store	ST	Transfers data from a processor register to a memory location
Move	MOV	Transfers data between register-register OR between Memory-Register
Exchange	XCH	Swaps data between register-register OR between Memory-Register
Input	IN	IN read from a part.
Output	OUT	OUT write to part.
Push	PUSH	Pushes the data from a register onto a memory stack
Pop	POP	Pops the data from a memory stack and store it onto a register

Mnemonics are symbols used to represent various instructions. An instruction can be specified in various addressing modes. As a specific case, the immediate mode is recognized from a pound sign '#' placed before the operand. For example the LOAD to AC instruction can be used in eight different addressing modes as follows:

Eight Addressing Modes for the Load Instruction

Mode	Assembly Convention	Register Transfer
Direct address	LD ADR	AC ← M[ADR]
Indirect address	LD @ADR	AC ← M[M[ADR]J
Relative address	LD $ADR	AC ← M[PC + ADR]
Immediate operand	LD #NBR	AC ← NBR
Index addressing	LD ADR(X)	AC ← M[ADR + XR]
Register	LD R1	AC ← R1
Register indirect	LD (R1)	AC ← M[R1]
Autoincrement	LD (R1) +	AC ← M[R1], R1 ← R1 + 1

Abbreviations used has meaning as follows : ADR=address, NBR=number/operand, X=index register, AC=Accumulator, R1=General purpose register, @=indirect addressing mode, $=relative address to the PC,

2. Data Manipulation Instructions:

Data manipulation instructions perform various operations on data and provide the computational capabilities for the computer. We can further classify the Data Manipulation Instructions in following 3 types :

I. Arithmetic Instructions:

It includes addition, subtraction, multiplication, and division. Following table shows various arithmetic instructions in a typical computer:

Typical Arithmetic Instructions

Name	Mnemonic
Increment	INC
Decrement	DEC
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Add with carry	ADDC
Subtract with borrow	SUBB
Negate (2's complement)	NEG

Overflow and Underflow :

If increment operation is applied to an n-bit register whose all bits contain '1', the result after increment operation would not fit into that register. After increment all bits of the register will contain '0'.This is called overflow of the data.

(Overflow of the data)

Similarly, If decrement operation is applied to an n-bit register whose all bits contain '0', the operation needs a borrow that is not available due to all 0s. After decrement all bits of the register will contain '1s'. This is called underflow of the data .

(Underflow of the data)

	1111	1111	1111	1111
				+1
1	0000	0000	0000	0000
1	0000	0000	0000	0000
				-1
	1111	1111	1111	1111

The arithmetic instructions may have different variants on a typical computer. For example the "ADD" instruction may have following 3 variants : For binary integers, For floating-point operands, and For decimal operands. Carry resulting

from any arithmetic instruction (like as discussed in OVERFLOW) is stored in a special "Carry Flip-Flop" that enables the system to use it for the instruction "Add with Carry" that performs the addition of two operands along with the previous carry. Similarly, the "subtract with borrow" instruction subtracts two words and a borrow which may have resulted from a previous subtract operation. The negate instruction forms the 2's complement of a number, effectively reversing the sign of an integer when represented in the signed-2's complement form.

II. Logical and Bit Manipulation Instructions:

Logical instructions include bitwise AND, bitwise OR etc. Logical and bit manipulation instructions in a typical computer are as follows :

Typical Logical and Bit Manipulation Instructions

Name	Mnemonic
Clear	CLR
Complement	COM
AND	AND
OR	OR
Exclussive-OR	XOR
Clear carry	CLRC
Set carry	SETC
Complement carry	COMC
Enable interrupt	EI
Disable interrupt	DI

The CLR instruction replaces all the bits of the operand (register) with 0. The COM (Complement) instruction exchanges 0s with 1s and vice-versa. The AND, OR, and XOR instructions produce the corresponding logical operations on individual bits of the operands.

Truth tables explains AND, OR and XOR operations :

Bit	Not Bit
0	1
1	0

Example :
MOV BL, 01010111B
AND BL, 11110000B

Result in BL : 01010000B

R1	R2	R1 AND R2	R1 OR R2	R1 XOR R2
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Other bit manipulation instruction has following effects :

- **Clear Carry (CLRC) :** clear the carry bit to 0
- **Set Carry (SETC) :** Sets the carry bit to 1
- **Complement Carry(COMC) :** inverts the value of the carry bit
- **Enable Interrupt :** enables the interrupt mode
- **Disable Interrupt :** disables the interrupt mode

III. Shift Instructions:

These instructions are used to move the bits of a Memory Word or register in a particular directions. Shift instructions have many variants like : logical shifts, arithmetic shifts, or circular shifts. In either case the shift may be to the right or to the left. Various shift instructions in a typical computer are as follows :

Typical Shift Instructions

Name	Mnemonic
Logical shift right	SHR
Logical shift left	SHL
Arithmetic shift right	SHRA
Arithmetic shift left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right through carry	RORC
Rotate left through carry	ROLC

(a) Logical Shifts :

These instructions insert 0 at the last bit position and shift all the bits of the memory word in specified direction by one bit position.

(b) Arithmetic shifts :

These instructions are meant for signed-2's complement numbers. These instructions preserve the sign bit in the leftmost position.

In case of Arithmetic shift right(SHRA), while shifting of individual bits , the sign bit is also shifted to the right together with the rest of the number, but the sign bit itself remains unchanged. In case of arithmetic shift-left (SHLA) 0 is inserted at the last bit position and is identical to the logical shift-left instruction.

(c) Circular Shifts :

These instructions treat the bit sequence as a circular list. It means that the rightmost bit position is treated adjacent to the leftmost bit. Concept is similar to a circular queue, in which the elements that get deleted from one end of the queue inserted at the other end of the queue. Another variants of circular shifts are RORC and ROLC. A rotate-left through carry (RORC) instruction transfers the carry bit into the rightmost bit position of the register, transfers the leftmost bit position into the carry, and at the same time, shifts the entire register to the left. Similar concept could be applied to the ROLC as well.

Example :

1. Logical Shift Right:

Before shift : 1001 1010 0110 1111

After Shift : **0**100 1101 0011 0111

2. Logical Shift left :

Before shift : 1001 1010 0110 1111

After Shift : 0011 0100 1101 1110

Following examples will make circular shifts more clear :

Before Shift	1	0	0	1	0	0	1	**0**
After Rotate right (ROR)	**0**	1	0	0	1	0	0	1
After Rotate left (ROL)	0	0	1	0	0	1	0	1

Before Shift	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**Carry = 0**
After RORC	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**0**
After ROLC	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**1**

ADDRESSING MODES

The opcode field of an instruction specifies the operation to be performed e.g. addition, subtraction, complement etc. Most instructions need operands, so we need to specify where the operands are in memory for an instruction. This is called the addressing of the data. Addressing mode refers to the system how the bits of an address field are interpreted to find the operands. In other words the way the operands are

A typical instruction has the following format :

Opcode	Address Field

chosen during program execution is referred to as the addressing mode of the system. The address field in a typical instruction format is relatively small. We need to reference a large range of locations in main memory. For this the no. of bits in address field are sometimes insufficient. For example if address field has 7 bits then it can only differentiate between 2^7 different addresses. If the main memory has more than 2^7=128 words, then we need different approach to address all words of the memory. To achieve this different addressing schemes have been employed. Following addressing schemes are commonly used:

1. Implied Mode
2. Immediate Mode
3. Direct Addressing
4. Indirect Addressing mode
5. Register mode
6. Register Indirect Addressing mode
7. Indexed Addressing

A single computer can use multiple addressing modes for different instructions. Here the question is, How does it differentiate between these addressing modes? Most common approach is to use one or more bits as "MODE FIELD" in instruction format. The value of the MODE FIELD determines that which addressing mode is to be used for a particular instruction.

A typical instruction format with mode field is given below :

Opcode	Mode Field	Address field

There bits in the address field may be interpreted as operand or location. If address is not required for an instruction then corresponding bits may designate an operand itself. If an instruction requires address, the address field may designate a memory address or a processor register.

Although most addressing modes need the address field of the instruction, there are two modes that need no address at all. These are the "implied mode" and "immediate mode".

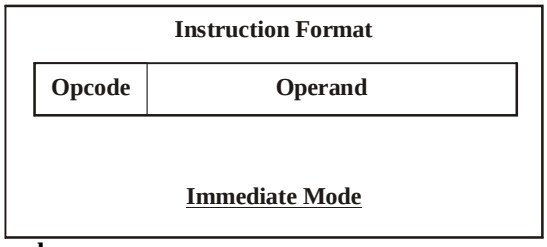
1. **Implied Mode:**

   There are some instructions that need no operand at all. For such instructions the operands are interpreted automatically by the definition of the instruction itself. For example, the instruction "Complement AC" is an implied mode instruction because the data in the AC register is to be complemented and no need to specify where the data resides. All register reference instructions that use Accumulator (AC) are implied mode instructions. Zero address instructions in a computer based on stack organization are also implied mode instructions where the operands are always fetched from the top of the stack.

Example :

MOV	R2	100

(An Immediate instruction for loading a constant "100" into register R2)



2. **Immediate Mode :**

   Many times the instructions need constant operands. Rather than specifying addresses, the operand itself can be provided at the address field of the instructions. Such scheme is called Immediate Mode addressing. This is automatically fetched from memory when the instruction itself is fetched.

   Since the operand is directly available in the instruction itself, no extra memory reference is needed to fetch the operand, thereby reducing the execution time. However, the disadvantage is that only a constant can be supplied as an operand using this method. Also the value of the constant is limited by the size of the address field.

Example :

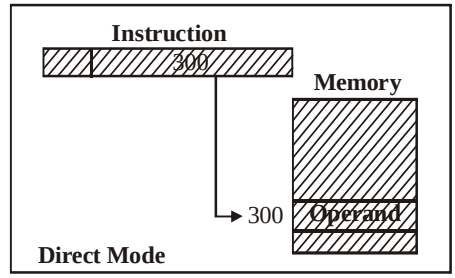
MOV	R2	0000 0001 0010 1100

(Above instruction fetches the data at address "0000 0001 0010 1100" (decimal: 300) from main memory and loads it in the register R2)

3. **Direct Addressing:**

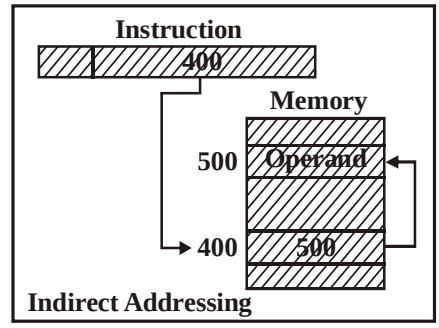
   Another method for specifying an operand in memory is by giving its full address of main memory.

This mode is called direct addressing. Direct addressing is restricted in its use: Although, the address value can be changed, it can only be used to access global variables whose address is known at compile time.



4. Indirect Addressing Mode :

In this mode that address is stored at the address field of the instruction where the address of the operand is located. For Example, Let the operand be located at address 500. Now there is another memory word at address 400 whose value (i.e. data stored at it) is 500. So, in Indirect addressing mode the value 400 will be stored at the address field of the instruction. Following figure will make it more clear :

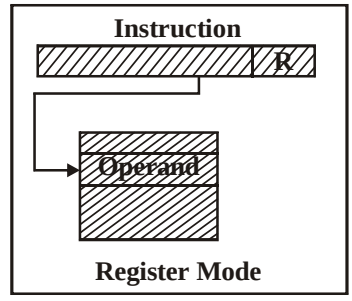


5. Register Addressing :

Register addressing is similar to the above discussed direct addressing. The difference between both is that in this case the address field specifies a register instead of a memory location. Because registers are so important (due to fast access and short addresses) this addressing mode is the most common one on most computers. This is often used by modern compilers for code optimaztion e.g. To store those variables of loop that are accessed frequently (e.g. loop counters). This addressing mode is known simply as register mode.

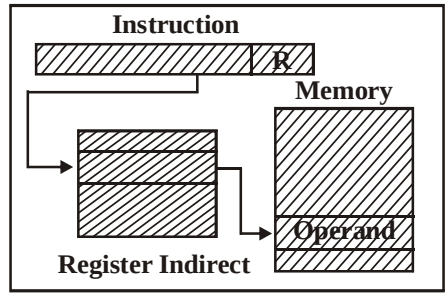
6. Register Indirect Addressing :

This mode is similar to Indirect addressing mode discussed above. In this mode also, the operand being specified comes from memory or goes to memory, but its address provided in the instruction is not absolute as in direct addressing. Instead, the address is contained in a register which is identified by the address field of the instruction. So, in other words the operands are found at some address and this address is stored in a register which is identified by the address field of the instruction.



7. Indexed Addressing :

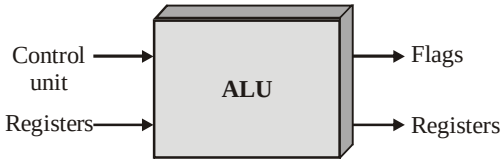
In many cases it is good to reference memory words at a known offset from an address stored in register. Addressing memory by giving a register (explicit or implicit) plus a constant offset is called indexed addressing. This is specially useful while accessing elements of an array in a loop. To see how that works, consider the following example. We have One one-dimensional array 'X' of 500 numbers, and we wish to find largest number in the array. In this case we can put the base address of X in one register R, and then step through its elements one by one by adding fixed constant offset to address stored in register R. The fixed constant is equal to the size of one element of the array.



All addressing modes with Examples

Addressing mode	Example instruction	Meaning	When used
Immediate	Add R4, #3	Regs [R4] ← Regs [R4] + 3	For constants.
Direct or absolute	Add R1, (1001)	Regs [R1] ← Regs [R1] + Mem [1001]	Sometimes useful for accessing static data; address constant may need to be large.
Memory indirect	Add R1, @(R3)	Regs [R1] ← Regs [R1] + Mem[Mem[Regs[R3]]]	If R3 is the address of a pointer P, then mode yields *p*.
Register	Add R4, R3	Regs [R4] ← Regs [R4] + Regs[R3]	When a value is in a register.
Register indirect	Add R4, (R1)	Regs [R4] ← Regs [R4] + Mem[Regs[R1]]	Accessing using a pointer or a compound address
Indexed	Add R3, (R1 + R2)	Regs [R3] ← Regs [R3] + Mem[Regs[R1] + Regs [R2]]	Sometimes useful in array addressing: R1 = base of array; R2 = index amount

ALU (ARITHMETIC AND LOGICAL UNIT)

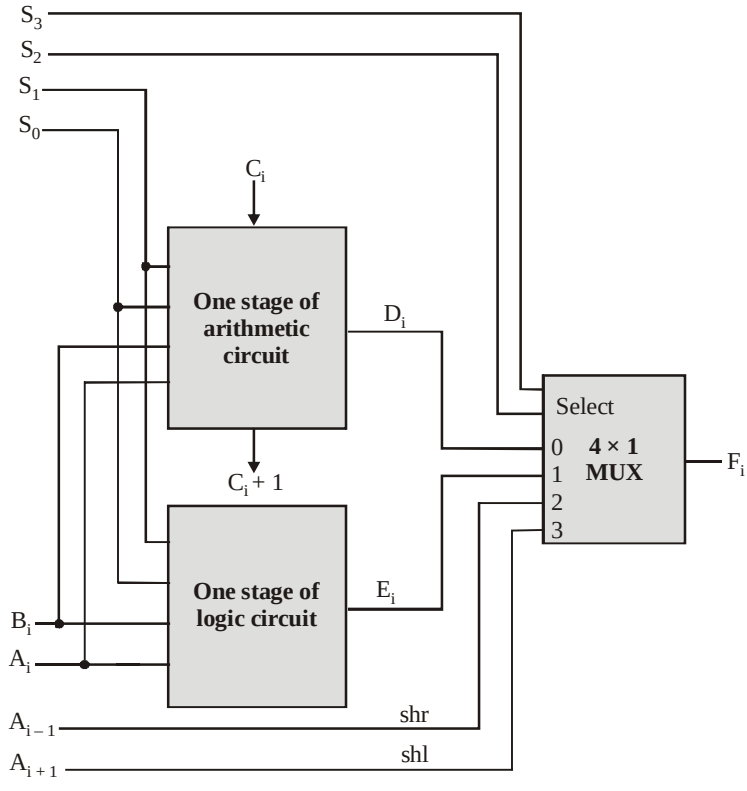


ALU, the Arithmetic and Logical Unit of the Central Processing Unit is that part of the computer that actually performs arithmetic and logical operations on data. Other parts of the system (like Control Unit, Registers, Memory and I/O devices), hosts the data that is brought to the ALU for computation. ALU performs the required operation on the data and then the results are taken back to other parts of the system. ALU contains various electronic circuits required to perform various Arithmetic (e.g addition, subtraction etc) and Logical (e.g. AND, OR, Complement etc) operation.

Following block diagram shows the Input and Output of an ALU :

Data are presented to the ALU in registers, and the results of the operation are also stored in registers. To perform a microoperation the contents of specified registers are placed in the inputs of the ALU. ALU performs the required microoperation and the result is then transferred to a destination register. These registers are temporary storage locations, within the processor that are connected by signal paths to the ALU. The control unit provides signals that control the operation of the ALU and the movement of the data into and out of the ALU. There are various control variable called "flags" that can be set by the ALU after it performs the required operation on the data. For example an overflow flag can be set by ALU if the result of an operation has bigger size then a register can hold. The registers that hold the operands, the registers that hold the result of the operation performed by the ALU and the registers that hold the "flags", All resides within the CPU.

One stage of arithmetic logic shift unit



The ALU shown in figure has two components : One to perform Arithmetic operation and other to perform Logical operations. D_i is output of Arithmetic subunit while E_i is output of logical subunit of ALU. Inputs A_i and B_i are supplied to the both components of the ALU. Inputs S_0 and S_1 decides what micooperation is to be performed on the data. A 4 x 1 multiplexer is used to choose one of the arithmetic output D_i and Logical Output E_i. The data in the multiplexer are selected with inputs S_3 and S_2. In case of shift-right operation and shift-left operation the other two data inputs A_{i-1} (for the shift-right operation) and A_{i+1} (for the shift-left operation) need to be supplied to multiplexor.

The shown figure is a simplification of the actual ALU. This shows the part of the ALU handling only 1-bit of information. Above figure must be repeated N-times for an N-bit ALU. Moreover, The output carry C_{i+1}, of a given arithmetic stage must be connected to the input carry C_i, of the next stage. The input carry to the first stage is the input carry C_{in}, which provides a selection variable for the arithmetic operations. The above circuit provides eight arithmetic operation, four logic operations, and two shift operations. Each operation is selected with the five variables S_3, S_2, S_I, S_o, and C_{in}, The input carry C_{in} is used for selecting an arithmetic operation only. A and B are two four bit input lines which is used to select a particular operation in the unit. The selection lines are decoded within ALU. F is output, S2 is mode select. It is set to 0 for Arithmetic operation and 1 for logical operations. C in and C out to carry input and output lines.

Following Table lists all the 14 operations of the ALU. The first eight are arithmetic operations and are selected with S_3=0, S_2=0. The next four are logic operations and are selected with S_3=0, S_2=1. The input carry has no effect during the logic operations and is marked with don't-care x's. The last two operations are shift operations and are selected with S_3=1, S_2 = 0 and S_3=1, S_2 =1. Other three selection inputs S_1, S_0 and C_{in} have no effect on the shift operations.

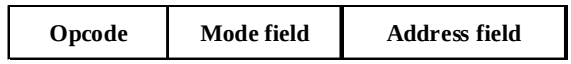
Funtion Table for Arithmetic Logic Shift Unit

Operation select						
S_3	S_2	S_1	S_0	C_{in}	Operation	Function
0	0	0	0	0	F = A	Transfer A
0	0	0	0	1	F = A + 1	Increment A
0	0	0	1	0	F = A + B	Addition
0	0	0	1	1	F = A + B + 1	Add with carry
0	0	1	0	0	F = A + $\overline{B}$	Subtract with borrow
0	0	1	0	1	F = A + $\overline{B}$ + 1	Subtraction
0	0	1	1	0	F = A – 1	Decrement A
0	0	1	1	1	F = A	Transfer A
0	1	0	0	×	F = $A \wedge B$	AND
0	1	0	1	×	F = $A \vee B$	OR
0	1	1	0	×	F = $A \oplus B$	XOR
0	1	1	1	×	F = $\overline{A}$	Complement A
1	0	×	×	×	F = shr A	Shift right A into F
1	1	×	×	×	F = shl A	Shift left A into F

CPU CONTROL DESIGN

Design of Control Unit

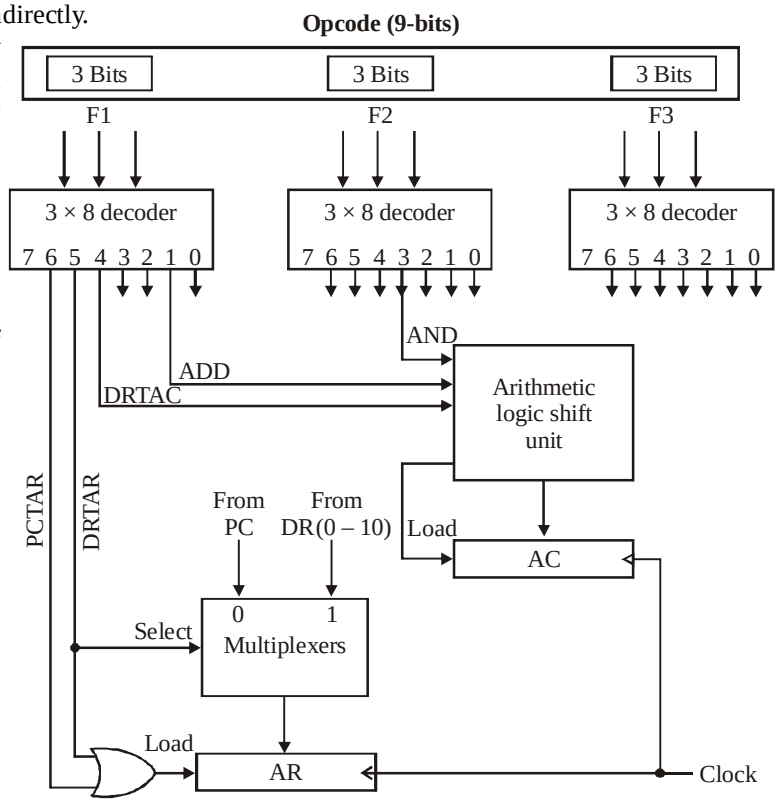
The bits of the microinstructions consist of various fields. These fields include opcode field, mode field and address field. The opcode field provides control bits to perform some microoperations like addition, complement etc. in the system. The mode field bits specify the addressing mode using which the address field is to be interpreted. The address field is used to specify the operand, or the address of the operand either directly or indirectly.



(A Typical Instruction Format)

The number of bits in the Opcode field can be reduced by grouping mutually exclusive variables into fields and encoding the k bits in each field to provide 2^k microoperations. Each field requires a decoder to produce the corresponding control signals. This method requires additional hardware outside to the control memory. As the signals passes through this additional hardware, it increases the delay time of the control signals.

The nine bits of the Opcode field are divided into three subfields of three bits each. The control memory output of each subfield must be decoded to provide the distinct microoperations. The outputs of the decoders are connected to the appropriate inputs in the processor unit. Above Figure shows the three decoders. Each of the three fields of the microinstruction presently available in the output of control memory are decoded with a 3 x 8 decoder to provide eight outputs. Each outputs is connected to the proper circuit to initiate the corresponding microoperation As shown in above Fig, outputs 5 and 6 of decoder F1 are connected to the load input of AR so that when either one of these outputs is active, information from the multiplexers is transferred to AR. The multiplexers select the information from DR when output 5 is active and from PC when output 5 is inactive. The transfer into AR occurs with a clock pulse transition only when output 5 or output 6 of the decoder is active.

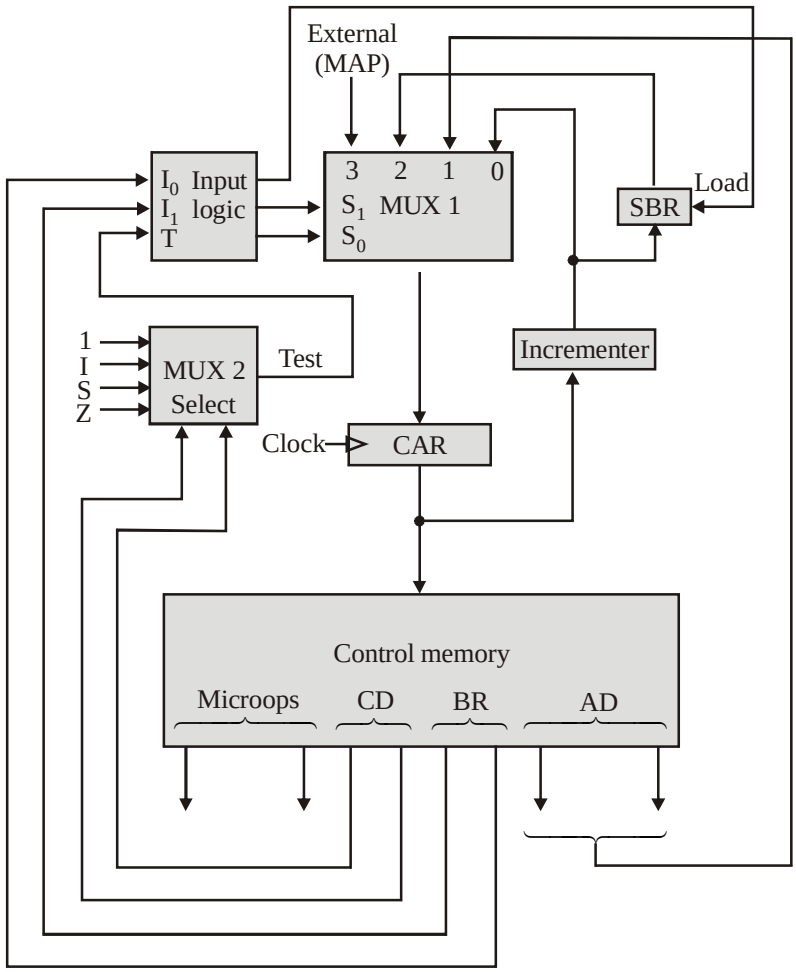


Decoding of microoperation fields

The other outputs of the decoders that initiate transfers between registers must be connected in a similar fashion. The control signals come from the outputs of the decoders associated with the symbols AND, ADD, and DRTAC, respectively. The other outputs of the decoders that are associated with an AC operation must also be connected to the arithmetic logic shift unit in a similar fashion.

Microprogram Sequencer

A Microprogrammed control unit mainly consists of control memory and the circuits that select the next address. The address selection part is called a microprogram sequencer.



Microprogram sequencer for a control memory

The purpose of a microprogram sequencer is to present an address to the control memory so that next microinstruction may be fetched from memory and executed. The next-address logic of the sequencer determines the specific address source to be loaded into the control address register. The present microinstruction supplies the information to the microsequencer about how the next address will be selected. Some sequencers provide an output register which can function as the address register for the control memory. The internal structure of a microsequencer is shown in figure below :

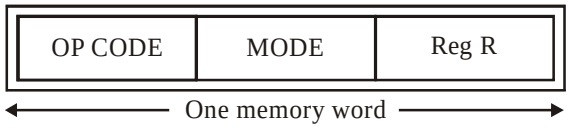
The control memory is included in the diagram to show the interaction between the sequencer and the memory attached to it. The circuit contains two multiplexors. The first multiplexer selects an address from one of four sources and routes it into a control address register CAR. The second multiplexer tests the value of a selected status bit and the result of the test is applied to an input logic circuit. The address for the control memory is provided at the output of CAR. The content of CAR is incremented and applied to one of the multiplexer inputs and to the subroutine register SBR. The other three inputs to multiplexer number 1 come from the address field of the present microinstruction, from the output of SBR, and from an external source that maps the instruction. The CD (condition) field of the microinstruction selects one of the status bits in the second multiplexer. If the bit selected is equal to 1, the T (test) variable is equal to 1; otherwise, it is equal to 0. The T value together with the two bits from the BR (branch) field goes to an input logic circuit. Typical sequencer operations are: increment, branch or jump, call and return from subroutine, load an external address, push or pop the stack, and other address sequencing operations. With three inputs, the sequencer can provide up to eight address sequencing operations. The input logic circuit in above figure has three inputs, I_0, I_1, and T, and three outputs, S_0, S_1, and L. Variables S_0 and S_1 select one of the source addresses for CAR. Variable L enables the load input in SBR. The binary values of the two selection variables determine the path in the multiplexer. For example, with S_1 S_0 = 10, multiplexer input number 2 is selected and establishes a transfer path from SBR to CAR. Note that each of the four inputs as well as the output of MUX 1 contains a 7-bit address. The following is the truth table for the input logic circuit:

BR Field		Input I_1	I_0	I_T	MUX 1 S_1	S_0	Load SBR L
0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0
0	1	0	1	0	0	0	0
0	1	0	1	1	0	1	1
1	0	1	0	×	1	0	0
1	1	1	1	×	1	1	0

Input Logic Truth Table for Microprogram Sequencer

Inputs I_1 and I_0 are identical to the bit values in the BR field. The bit values for S_1 and S_o are determined from the stated function and the path in the multiplexer that establishes the required transfer. The subroutine register is loaded with the incremented value of CAR during a call microinstruction (BR = 01) provided that the status bit condition is satisfied (T = 1).

Example 1: The instruction format of a CPU is



Mode and Reg R together specify the operand. Reg R specifies a CPU register and Mode specifies an addressing mode. In particular, Mode = 2 specifies that the register Reg R contains the addrress of the operand, after fetching the operand, the contents of Reg R are incremented by 1'.

An instruction at memory location 2000 specifies Mode = 2 and Reg R refers to program counter (PC).

(i) What is the address of the operand ?

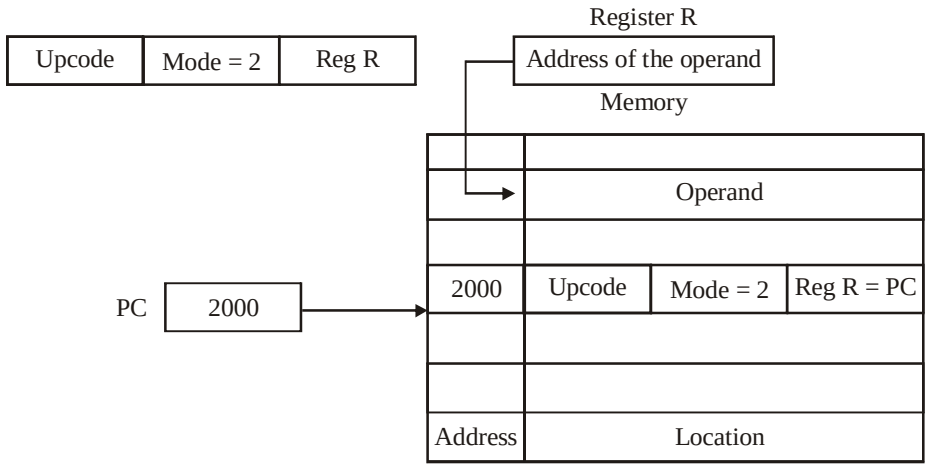
(ii) Assuming that this is a non-jump instruction, what are the contents of PC after the execution of the instruction ?

Sol.

(i) Address of the operand is the content of PC

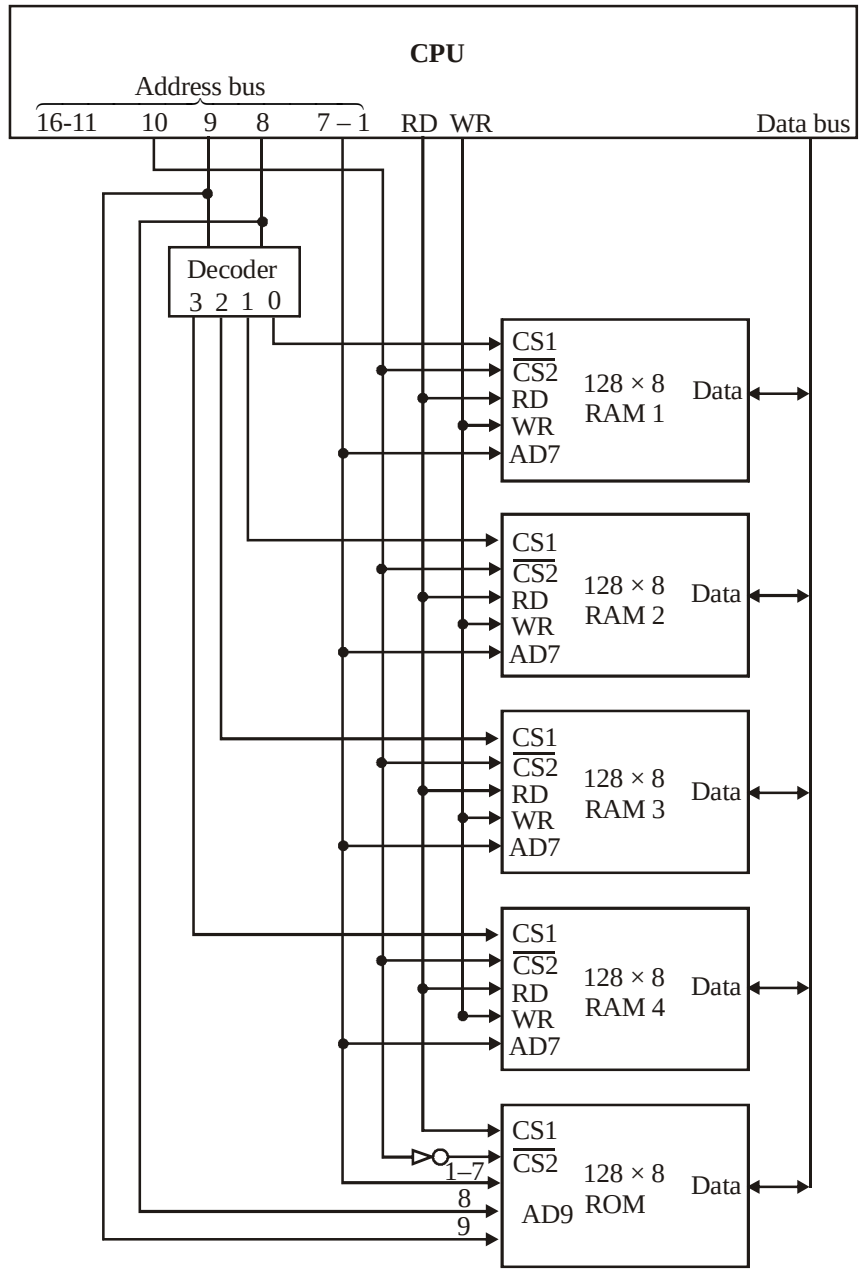
(ii) 2001

Method :



MEMORY INTERFACE

The data and address buses connect memory chips (RAM and ROM) to the CPU. Some bits in the address bus are used to select one of the many memory chips while other bits select the memory word in the selected chip. Memory chips may be connected to the CPU as follows :



Memory connection to the CPU

This configuration gives a memory capacity of 512 bytes (128x4) of RAM and 512 bytes of ROM.

The low order bits (1-7) in the address bus are used to select one of 128 possible bytes addresses.

The bits 8 and 9 in the address bus are used to select a particular RAM chip using a 2x4 decoder. The output of the decoder is connected to CS1 input of every RAM chip. Following table shows how a particular RAM chip is selected based on values of 8^{th} and 9^{th} bits in the address bus :

8th bit in address bus	9th bit in address bus	RAM chip Selected
0	0	1st RAM Chip
0	1	2nd RAM Chip
1	0	3rd RAM Chip
1	1	4th RAM Chip

Address bus lines 1 to 9 are applied to the input address of ROM without going through the decoder. The 10th bit in the address bus is used to decide that operation is performed on RAM or ROM. It is clear from above figure that if 10th bit is 0, inputs in all RAM chips will become 0 and all RAM Chips will be enabled, however input in ROM chip will become 1 and it will be in high impedence state. In other case if 10th bit is 1, inputs in all RAM chips will become 1 and all RAM Chips will be disabled, however input in ROM will become 0 and ROM will be enabled. This can be tabulated as follows :

Value of 10th bit in address bus	Active Chip
0	RAM
1	ROM

The RD (READ) and WR (WRITE) outputs that work as the control signal are applied at the inputs of each RAM chip. The CS1 input in the ROM is connected to the RD control line for the ROM chip to be enabled only during a READ operation.

The data bus of the ROM is unidirectional with only capability to perform READ operation (output data). On the other hand, the data bus connected to the RAMs is bidirectional i.e. it can transfer information both from CPU to Memory (Write Operation) and from Memory to CPU(READ Operation).

When the 10th bit is equal to 0, i.e. when RAM is enabled the operations on different inputs can be summarized by the following table :

CS1	$\overline{CS2}$	RD	WR	Memory function	State of data bus
0	0	×	×	Inhibit	High-impedance
0	1	×	×	Inhibit	High-impedance
1	0	0	0	Inhibit	High-impedance
1	0	0	1	Write	Input data to RAM
1	0	1	×	Read	Output data from RAM
1	1	×	×	Inhibit	High-impedance

When the 10th bit is equal to 1, i.e. when ROM is enabled the operations on different inputs can be summarized by the following table :

CS1	$\overline{CS\ 2}$	RD	Memory function	State of the data bus
0	0	X	Inhibit	High Impedance
0	1	X	Inhibit	High Impedance
1	0	0	Inhibit	High Impedance
1	**0**	**1**	**Read**	**Read data from ROM**
1	1	X	Inhibit	High Impedance

So We note that ROM is only enabled when CS1=1, $\overline{CS2}$ = 0 and RD input is enabled. READ Operation is performed on ROM in this case. ROM is disabled or data bus is in high impedence state in all other cases. We also note that when the chip is not enabled either CS1=0 or =1, the value of RD input becomes insignificant which is denoted by "X" in the above table.

INPUT/OUTPUT INTERFACE

External Devices

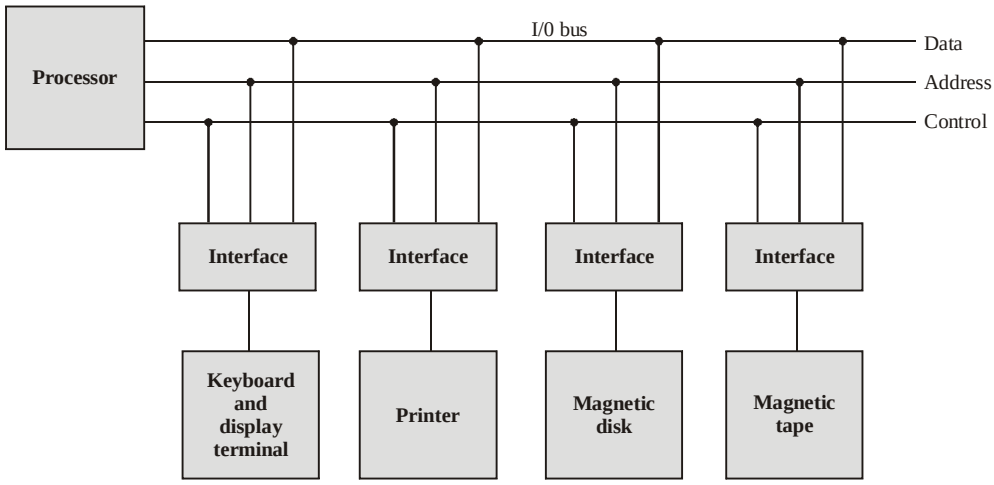
The input—output subsystem provides communication between the CPU and the external devices. A computer needs programs and data for processing and then it need to display the results obtained by processing. The most common way to feed data to the computer is through the keyboard. Other common external devices are display units, scanner, mouse, printers, magnetic disks and tapes etc.

Input—Output Interface

I/O interface provides a method for transferring information between CPU and external devices. The I/O interface resolves the differences that exist between the cpu and each External device. The major differences are:

1. Conversion of signals between CPU and external devices.
2. Synchronization issues due to the speed difference between CPU(extremely faster) and external devices (very slow).
3. Data codes and formats in External devices differ from the word format in the CPU and memory.
4. Difference in operating modes external devices with each other.

I/O interfaces are connected as shown in following figure :



I/O bus consists of data lines, address lines, and control lines. Each interface decodes the address and control received from the I/O bus, interprets them for the external device, and provides signals for the device controller along with synchronization. To communicate with a particular device, the processor places a device address on the address lines. When the interface detects its own address on the address line, it activates the path between the bus lines and the device that it controls. The CPU also provides a function code in the control lines. The selected interface responds to the function code (I/O Commands) and proceeds to execute it. I/O Commands are classified as control, status, data output, and data input. A control command is issued to activate the External device and to inform it what to do. A status command is used to test various status conditions in the interface and the external device. A data output command causes the interface to respond by transferring data from the bus into one of its registers. In case of data input command, the interface receives an item of data from the external device and places it in its buffer register. The processor checks if data are available by means of a status command and then issues a data input command. The interface places the data on the data lines, where they are accepted by the processor.

Isolated versus Memory-Mapped I/O

In the computers in which one common line is used to transfer data between memory or I/O and the CPU, a memory transfer and an I/O transfer is distinguished by separate read and write lines. The CPU enables one of two possible read or write lines to specify whether the address on the address lines is for a memory word or for an interface register. The I/O read and I/O write control lines are enabled during an I/O transfer. The memory read and memory write control lines are enabled during a memory transfer. In this system all I/O interface addresses are isolated from the memory addresses, so it is referred to as

the **isolated I/O method.** The isolated I/O method isolates memory and I/O addresses so that memory address values are not affected by interface address assignment since each has its own address space.

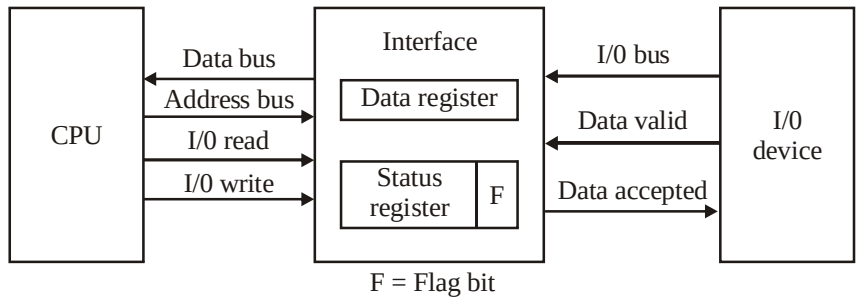
In the **Memory-Mapped organization,** same address space is used for both memory and I/O. The computer treats an interface register as being part of the memory system. The assigned addresses for interface registers cannot be used for memory words, which reduce the memory address range available. In a memory-mapped I/0 organization there are no specific input or output instructions. The CPU can manipulate I/O data residing in interface registers with the same instructions that are used to manipulate memory words. So, the computers with memory-mapped I/O can use memory-type instructions to access I/O data. It allows the computer to use the same instructions for either input—output transfers or for memory transfers. For example, the load and store instructions used for reading and writing from memory can be used to input and output data from I/O registers. This reduces the size of the instruction set of the computer.

Modes of data Transfer

Data transfer between CPU and I/O devices can be done in many ways which are called modes of data transfer. Following modes are used in a typical computer :

1. **Programmed I/O :**

Programmed I/O operations are based on the concept of constant monitoring of the external devices by the CPU. In the programmed I/O method, the CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer. Disadvantage is that it **keeps the processor busy needlessly.**



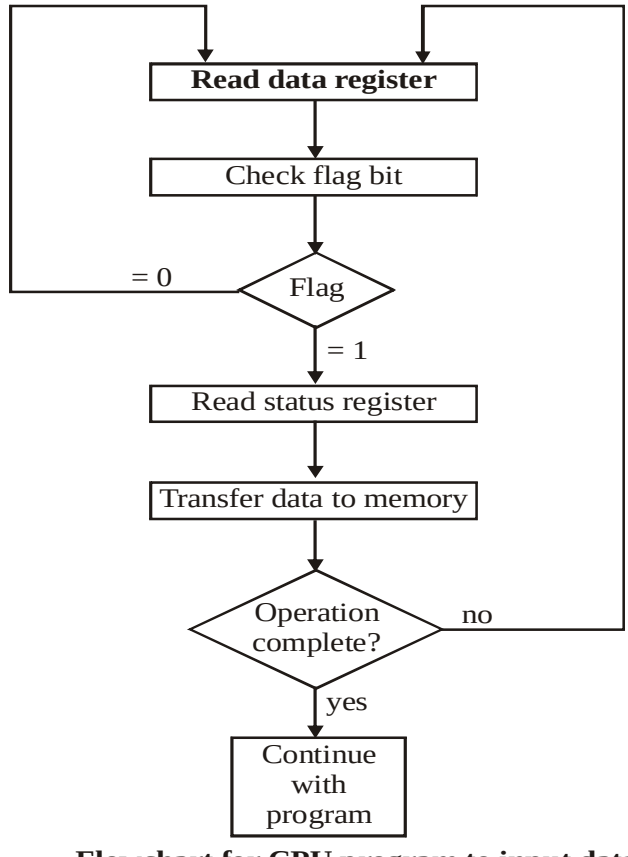
F = Flag bit

Example :

When a byte of data is available, the device places it in the I/O bus and enables its data valid line. The interface accepts the byte into its data register and enables the data accepted line. The interface sets a bit in the status register that we will refer to as an **F or "flag" bit.** The device can now disable the data valid line, but it will not transfer another byte until the data accepted line is disabled by the interface. This is done by reading the status register into a CPU register and checking the value of the flag bit. If the flag is equal to 1, the CPU reads the data from the data register. The flag bit is then cleared to 0 by either the CPU or the interface, depending on how the interface circuits are designed. Once the flag is cleared, the interface disables the data accepted line and the device can then transfer the next data byte.

2. **Interrupt-Initiated I/O:**

In this mode, the interface keeps monitoring the device and informs the CPU when it is ready to transfer data. CPU does not constantly monitor the flag. The flag is set by the interface when it is ready to transfer data. When the flag is set, the CPU is interrupted from executing the current program and is informed that the flag has been set. The CPU goes to take care of the input or output transfer. After the transfer is completed, the CPU returns to the previous program to continue what it was doing before the interrupt. The CPU responds to the interrupt signal by storing the return address from the program counter into a memory stack and then control branches to a service routine



Flowchart for CPU program to input data

that processes the required I/O transfer. The branch address can be chosen by one of the two methods: **Vectored interrupt and Non-Vectored interrupt.** In vectored interrupt, the source that initiates the interrupt, supplies the branch information to the CPU. This information is called the **Interrupt Vector.** The interrupt vector is either the first address of the I/O service routine OR it is an address that points to a location in memory where the beginning address of the I/O service routine is stored. In Non-Vectored interrupt, the branch address is always assigned to a fixed location in memory.

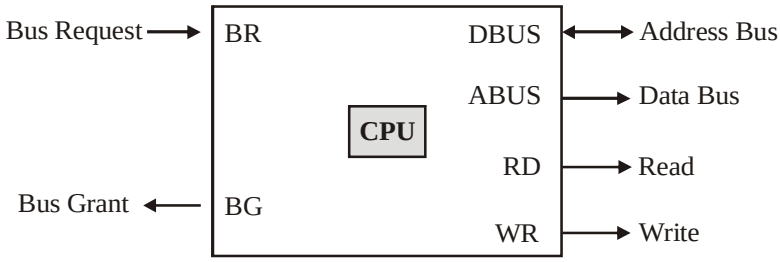
Priority Interrupt

In a typical computer system more than one I/O devices are attached to the computer, with each device being able to originate an interrupt request. The first task of the interrupt system is to identify the source of the interrupt. There is also the possibility that several sources will request service simultaneously. In this case the system must also decide which device to service first. A priority interrupt is a system that establishes a priority over the various sources to determine which interrupt is to be serviced first when two or more interrupt arrive simultaneously. The system may also determine which conditions are permitted to interrupt the computer while another interrupt is being serviced. Higher-priority interrupt levels are assigned to requests which, if delayed or interrupted, could have serious consequences. Devices with high-speed transfers such as magnetic disks are given high priority, and slow devices such as keyboards receive low priority. When two devices interrupt the computer at the same time, the computer services the device, with the higher priority first.

3. **Direct Memory Access:**

The disadvantage of above modes is that all data must go through the CPU. Although CPU is fast fast at computations it is relatively very slow when it comes to large amount of data transfer. So if data is transferred between main memory and I/O devices without going through CPU, high speed-up can be achieved, which is the essence of the **"Direct Memory Access (DMA)"** mode. In DMA, the I/O device controls the buses and directly transfers the data to and from main memory and CPU has no control over it. The CPU initiates the transfer by supplying the interface with the starting address and the number of words needed to be transferred and then proceeds to execute other tasks. When the transfer is made, the DMA requests memory cycles through the memory bus. When the request is granted by the memory controller, the DMA transfers the data directly into memory.

Diagram shows the CPU bus signals for DMA transfers :



Following Diagram shows the CPU bus signals for DMA transfers :

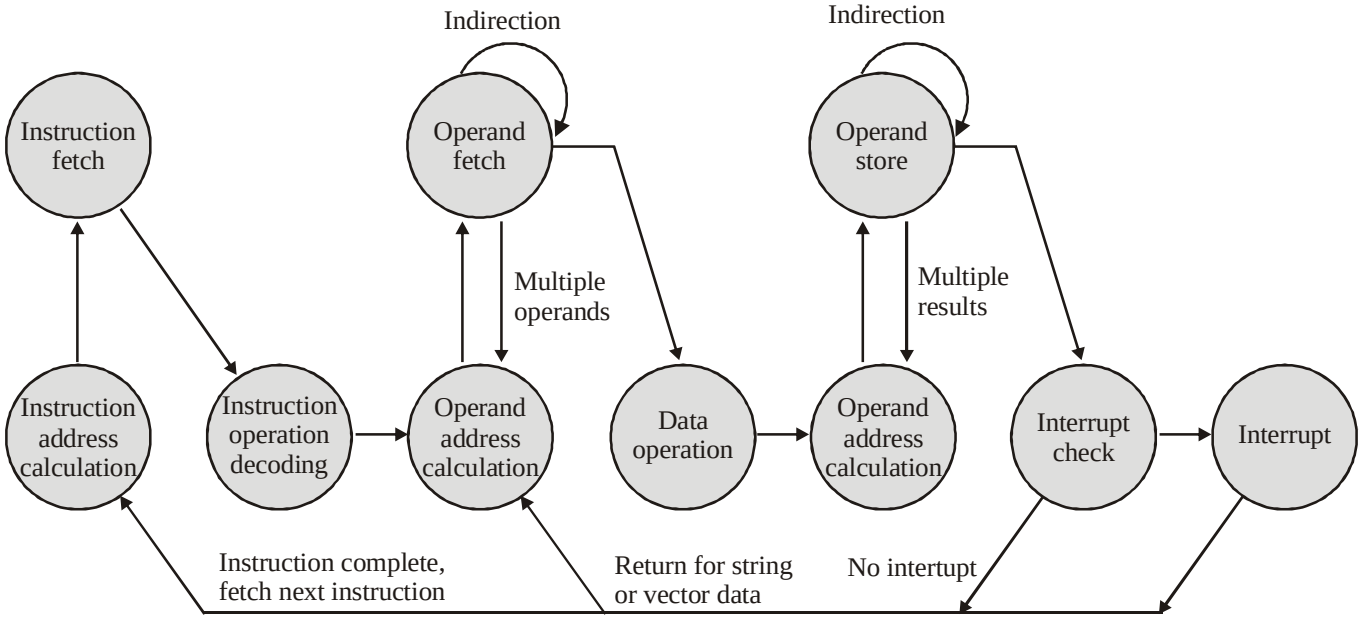
DMA controller requests the CPU to leave control of buses using BR input. On this request the CPU stops executing the current task and blocks Address bus, Data bus, Read line and write lines. Then CPU activates BG input to inform DMA controller that it can now use the buses. Now, the DMA controller takes over the control of all buses to perform data transfers between I/O devices and main memory. After the data transfer is completed, it disables the BR line. The CPU notes it and disables the BG line and takes over the control of all buses and resumes the execution where it stopped.

INSTRUCTION PIPELINE

There are various ways in which performance of a computer system can be improved. This includes using faster hardware, developing faster algorithms, code optimization by the compiler. Orgnizational approaches like Using multiple CPUs, Using multiple registers, Using various levels in memory hierarchy have also been used to improve the performance. Instruction pipelining is also an organizational approach which is commonly used in modern systems to improve the performance. Instruction pipelining is similar to the use

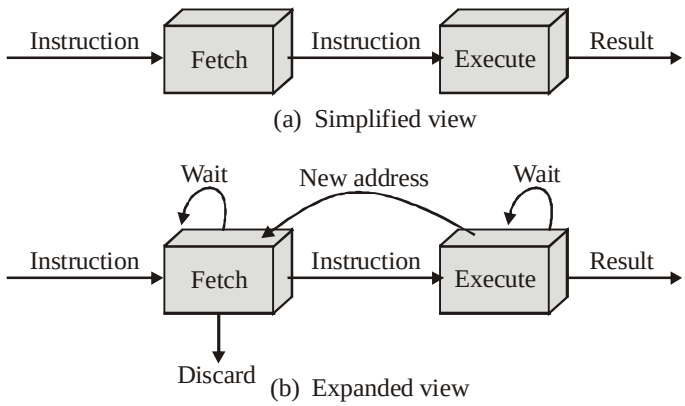
of an assembly line in a production factory. In an assembly line a product goes through various mutually exclusive stages of production. Products at various stages can be done simultaneously.

To apply this concept to instructions, we note that, in fact, an instruction has a number of stages.



Instruction Cycle State Diagram

In above figure the instruction cycle has been divided in separate tasks. This is similar to a factory assembly line. So we may apply the concept of pipelining. We can subdivide the instruction processing into two stages: fetch/decode and execute instruction. There are times during the execution of an instruction when main memory is not being accessed. This time could be used to fetch the next instruction in parallel with the execution of the current one. Consider the following figure :



Two-Stage Instruction Pipelines

The first stage fetches an instruction. In the next step, while the second stage is executing this instruction, the first stage takes advantage of any unused memory cycles to fetch the next instruction. If the fetch and execute stages were of equal duration, the instruction cycle time would be halved. However, there are two problem that may arise : firstly, the execution time will generally be longer than the fetch lime. Execution will involve reading and storing operands and then executing it. Thus, the fetch stage may have to wait for some time before it can proceed further. And secondly, a conditional branch instruction makes the address of the next instruction to be fetched unknown. In this case the fetch stage must wait until it receives the next instruction address from the execute stage

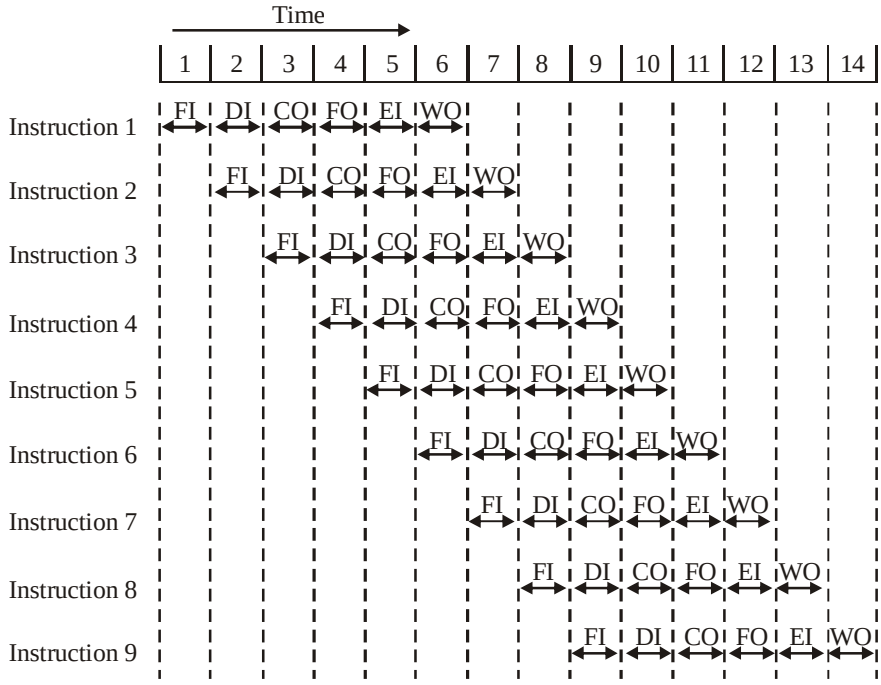
The execute stage may then have to wait while the next instruction is fetched. The common approach is to let the fetch stage get the next instruction in memory after the branch instruction. If the branch is not taken, no time is lost. If the branch is taken, the fetched instruction must be discarded and a new instruction fetched. While these factors reduce the potential effectiveness of the two-stage pipeline, some speedup occurs.

Now, if we further subdivide these two stages into smaller parts, we may achieve more speedup. Let us consider the following decomposition of the instruction processing.

- Fetch instruction (**FI :** Read the next instruction)
- Decode instruction (**DI :** decoding the opcode and operands specified)
- Calculate effective address of each operand (**CO :** it may include register indirect, indirect, or other forms of address calculation)

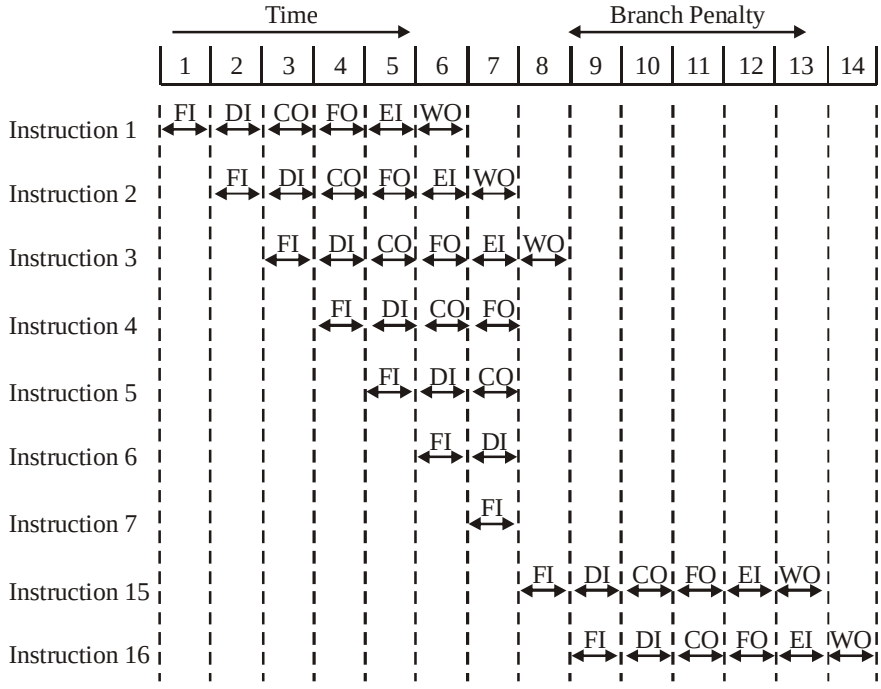
- Fetch operands from memory (**FO**)
- Execute instruction (**EI :** Perform the indicated operation and store the result)
- Write operand (**WO :** Store the result in memory)

This division leads to almost equal duration of various stages. Following Figure shows that a six-stage pipeline can reduce the execution time for 9 instructions from 54 time units to 14 time units.



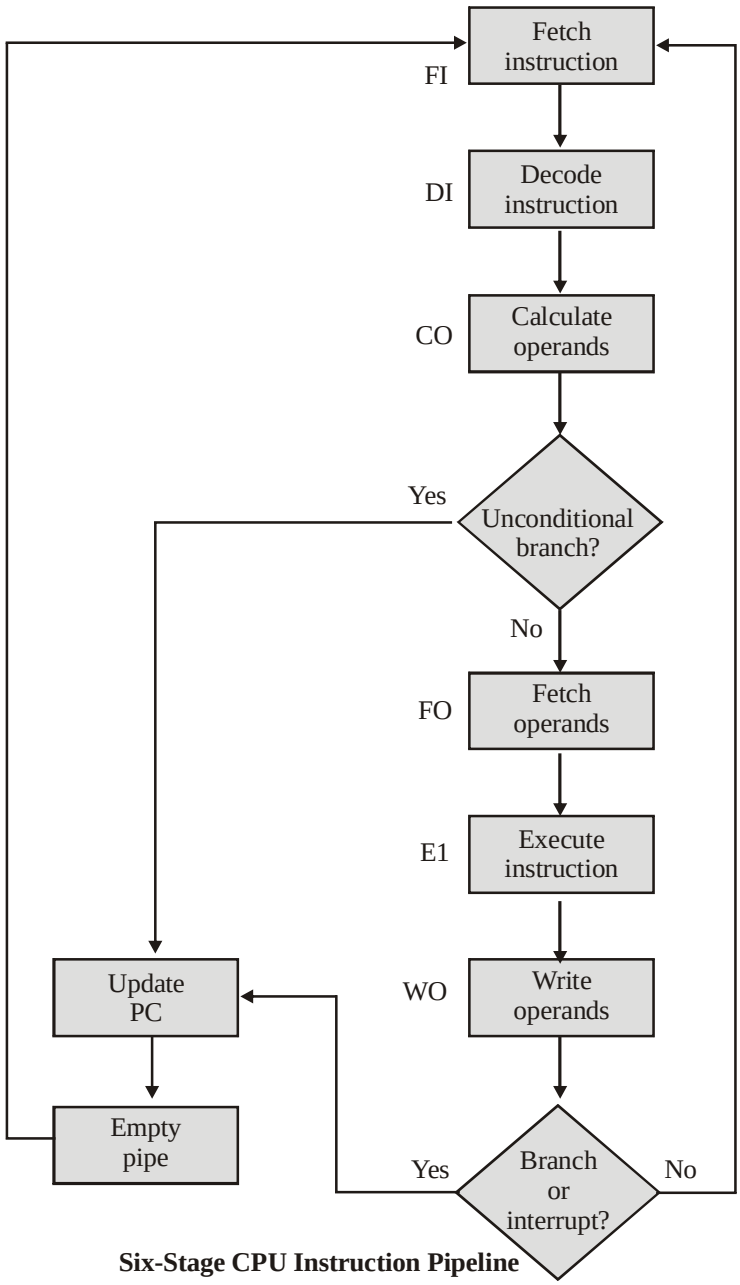
Timing Diagram for Instruction Pipeline Operation

If the six stages are not of equal duration, there will be some waiting involved at various pipeline stages, as discussed before for the two-stage pipeline. Another difficulty is the conditional branch instruction, which can invalidate several instruction fetches. Following Figure illustrates the effects of the conditional branch.



The Effect of a Conditional Branch on Instruction Pipeline Operation

In the Figure time diagram for instruction pipeline operation, assume that instruction 3 is a conditional branch to instruction 15. Until the instruction is executed, there is no way of knowing which instruction will come next. The pipeline, in this example simply loads the next instruction in sequence (instruction 4) and proceeds as discussed above. In Figure 3, the branch is not taken, and we get the full performance benefit of the pipelining. In-Figure 4, the branch is taken. This is not determined until the end of time t=7. Here, the pipeline is cleared of the instructions and at time t=8, instruction 15 enters the pipeline. No instructions complete during time units 9 through 12; this is the performance penalty incurred. Following flowchart indicates the logic needed for pipelining to account for branches and interrupts :



Six-Stage CPU Instruction Pipeline

So, Instruction pipelining is a powerful technique for enhancing performance but requires careful design to achieve optimum results with reasonable complexity.

Example 2: Consider the unpipelined machine with 10 ns clock cycles. It uses four cycles for ALU operations and branches whereas five cycles for memory operations. Assume that the relative frequences of these operations are 40%, 40% and 20%, respectively. Let due to clock skew and setup pipelining, the machine adds 1ns of overhead to the clock. How much speed in the instruction execution rate will gain from a pipeline ?

Sol. Average instruction execution time = Clock cycle × Average CPI

$$= 10 \text{ ns} \times [(40\% + 20\%) + 4 + 40\% \times 5]$$

$$= 10 \text{ ns} \times 4.4 = 44 \text{ ns}$$

In the pipelined implementation, clock must run at the speed of slowest stage plus overhead, which will be 10 + 1 or 11 ns; this is the average instruction execution time.

Thus, speed up from pipelining,

$$(\text{Speed up})_{\text{pipelinging}} = \frac{\text{Average instruction time unpiplined}}{\text{Average instruction pipelined}}$$

$$= \frac{44\text{ns}}{11\text{ns}} = 4 \text{ times}$$

Example 3: Assume that the time required for the five functional units, which operate in each of the five cycle are 10 ns, 8 ns, 10 ns, 10ns and 7 ns.

Assume that pipelining adds 1 ns of overhead. Find out the speed up versus single cycle data path

Sol. Since unpipelined machine executes all instructions in a single clock cycle, its average time per instruction is simply the clock cycle time. The clock cycle time is equal to sum of the times for each step in the execution.

$$\text{Average instruction execution time} = 10 + 8 + 10 + 10 + 7$$

$$= 45 \text{ ns}$$

Clock cycle time on the pipelined machine must be the largest time for any stage in the pipeline (10 nos) plus overhead of 1 ns, for a total of 11 ns.

Since CPI is 1, this yields an average instruction execution time of 11 ns.

$$\text{Speed from pipelining} = \frac{\text{Average instruction time unpiplined}}{\text{Average instruction time piplined}}$$

$$= \frac{45\text{ns}}{11\text{ns}} = 4.1 \text{ times}$$

Example 4: Consider a computer with four floating point pipeline processors. Let each processor uses a cycle time of 40 ns. How long will it take to perform 400 floating-point operations ? Is there a difference if the same 400 operations are carried out using a single pipeline processor with a cycle time of 10 ns ?

Sol. Divide 400 operations into each of the four processors,

$$\text{processing time} = \frac{400}{4} \times 40 = 4000 \text{ n sec}$$

Using a single pipeline,

$$\text{processing time} = 400 \times 10 = 4000 \text{ n sec}$$

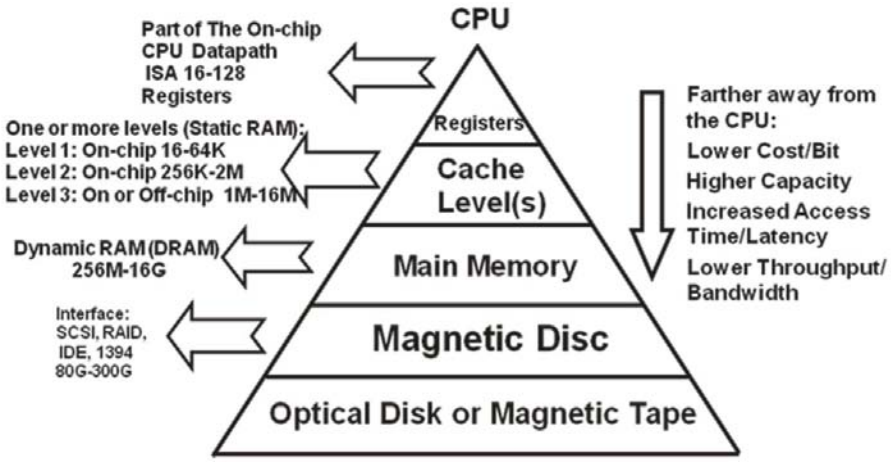
CACHE MEMORY

As we know CPU is much faster than memory. The problem comes into the picture when the CPU issues a memory request, it will not get the data it need for many CPU cycles. The slower the memory the more cycles the CPU will have to wait. This problem can be

overcome by introducing a small and very fast memory near the CPU. The small, fast memory is called **Cache Memory.** The basic idea behind a cache is simple: Keep those memory words in cache that are heavily required by CPU e.g. A loop counter. When the CPU needs a word, it first looks in the cache. Only if the word is not there does it go to main memory. If a substantial fraction of the words are in the cache, the average access time can be greatly reduced.

Memory access time is important to performance . Users want large memories with fast access times ideally unlimited fast memory.

Levels of the Memory Hierarchy



Memory Hierarchy

We can exploit the natural locality in programs by implementing the memory of a computer as a *memory hierarchy*.

Cache

Processor does all memory operations with cache.

- *Miss* - If requested word is not in cache, a *block* of words containing the requested word is brought to cache, and then the processor request is completed.
- *Hit* - If the requested word is in cache, read or write operation is performed directly in cache, without accessing main memory.
- *Block* - minimum amount of data transferred between cache and main memory.

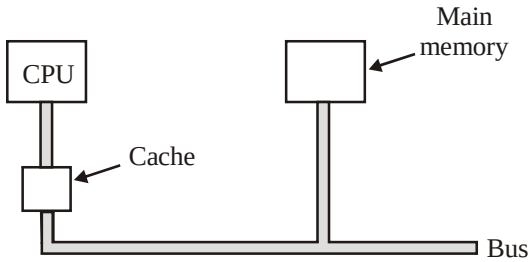
Temporal & Spatial Locality

There are two types of locality:

TEMPORAL LOCALITY (locality in time) If an item is referenced, it will likely be referenced again soon. Data is reused.

SPATIAL LOCALITY (locality in space) If an item is referenced, items in neighboring addresses will likely be referenced soon

Most programs contain natural locality in structure. For example, most programs contain *loops* in which the instructions and data need to be accessed repeatedly. This is an example of temporal locality. Instructions are usually accessed sequentially, so they contain a high amount of spatial locality. Also, data access to elements in an array is another example of spatial locality.



The general idea is that when a word is referenced, it and some of its neighbors are brought from the large main memory into the cache, so that the next time it is used, it can be accessed quickly. A common arrangement of the CPU, cache, and main memory is shown in fig.

So, if CPU finds it data in the cache it will work faster say for example if CPU accesses a data K times in a short period of time, if will need 1 reference to main memory and K-1 references to cache. The performance gets better with the larger values of K.

If the required memory word is found in cache, it is called a **"cache hit",** otherwise it is called a **"cache miss".**

CPU execution time must factor in stalls from memory access-

Assume the on-chip cache responds within the amount of time allotted to it from the load/store or instruction fetch unit (e.g., 1 clock cycle)

CPU time = IC * CPI * Clock cycle time

IC * CPI must include all stalls so now we have to add memory stalls

IC * CPI = CPU clock cycles + memory stall cycles

Where IC We can view memory cycle stalls as number of stalls per instruction or per memory access (loads & stores have 2 memory accesses per instruction, all others have 1 memory access per instruction)

Memory stall cycles = IC × (misses / instruction) × miss penalty

Memory stall cycles = IC × (memory accesses / instruction) × miss rate × miss penalty

Example 5 : A machine has CPI cycles per instruction 2.0 when all memory accesses hit in the chache. The only data accesses are loads and stores, and these total 40% of the instructions. If miss penalty is 25 clock cycles and miss rate is 2%, how much faster would be the machine be if all the instructions were cache hits ?

Sol. Performance for machine that always hits,

$$\text{CPU execution time} = (IC \times CPI + 0) \text{ clock cycle (where IC = instruction cycle)}$$
$$= IC \times 2.0 \times \text{clock cycle}$$

Now for machine with the real cache,

Memmory clock cycles

$$= IC \times \text{Memory reference per instruction} \times \text{miss rate} \times \text{Miss penalty}$$
$$= IC \times (1 + 0.4) \times 0.2 \times 25$$
$$= IC \times 0.7$$

where middle term (1 to 0.4) represents one instruction access and 0.4 data acesses per instruction.

Hence total performance is

$$(\text{CPU execution time})_{\text{Cache}} = (IC * 2.0 + IC * 0.7) * \text{clock cycle}$$
$$= 2.7 \times IC \times \text{clock cycle}$$

$$\frac{(\text{CPU execution time})_{\text{Cache}}}{\text{CPU execution time}} = \frac{2.7 \times IC \times \text{clock cycle}}{2.0 \times IC \times \text{clock cycle}} = 1.35$$

Hence machine with no cache misses is 1.35 times faster.

Hit Ratio: The ratio of the number of hits divided by the total CPU references to memory (hits plus misses) is the hit ratio.

Hit ratio = (no. of cache hits) / (Total no. of memory references)

A hit ratio of 0.8 means that 80 percent of total times the CPU would find the required memory word in cache itself.

Applying the locality principle, main memories and caches are divided up into fixed-size blocks. For cache memory these blocks are called "cache lines". When a cache miss occurs, the entire cache line is loaded from the main memory into the cache, not just the word needed. For example, with a 64-byte line size, a reference to memory address 260 will pull the line consisting of bytes 256 to 319 into one cache line. Now, some of the other words in the cache line will be needed shortly preventing further cache misses.

Example 6 : Let in a computer the cache has access time of 5 ns, main memory has access time of 20 ns, and the hit ratio is 90%.

Sol. Then the average access time is calculated as follows :

Average access time = (Hit ratio × Cache access time) + (1-Hit ratio) × (cache access time + main memory access time)

So, in above example, Average access time = 0.9 × 5ns + 0.1 × (5ns + 20ns) = 7 **ns**

Example 7 : When a cache is 10 times faster than main memory and cache can be used 70% of the time, how much speed we gain by using the cache ?

Sol. Let M = Main memory access time

C = Cache memory access time = M/10 (given)

Total access time using cache = $0.7 + 0.1\ M$

$$= 0.7\frac{M}{10} + 0.1M = 0.17\ M$$

$$\text{Speed up} = \frac{M}{0.19M} = 5.8$$

Three types of mapping functions are commonly used :

1. **Direct Mapping**
2. **Associative Mapping**
3. **Set-Associative Mapping**

Mappings Functions

As the size of cache is smaller it can occupy less number of words than main memory. So, we need a mapping function to select where a main memory block go in cache memory. We also need to ensure that the word can be searched in the cache in very less time. The transformation of data from main memory to cache memory is referred to as a mapping process.

For all three cases we consider the following configuration:

- Cache Block size=4 bytes and data can be transferred in multiples of blocks,
- Cache size = 64 KB, hence cache contains 2^{14} Blocks (called **"Cache Lines"**)
- Main memory block size = 4 bytes
- Main memory size = 16MB, hence main memory contains 2^{22} Blocks
- Each byte of main memory is directly addressable by a 24-bit address (as 2^{24} = 16M)

1. Direct Mapping :

In this technique each block of main memory can only go to a single fixed position in cache. A single block of cache is a possible candidate for few fixed blocks of main memory.

This mapping is expressed as :

i = j modulo m *where,* i = cache block number

j = main memory block number

m = number of blocks in cache

This mapping can be implemented by dividing the 24-bits of address into three parts. The least significant w-bits are used to identify a word within a block. The remaining s-bits are used to identify a main memory block among 2^s blocks. This field of s-bits is again divided into two parts for cache : r-bits line field(least significant) and s-r bits tag field (most significant). The line field identifies one cache line among 2^r lines of cache.

- Address length = (s + w) bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^w words or bytes
- Number of blocks in main memory = $\frac{2^{s+w}}{2^w} = 2^s$ words or bytes
- Number of lines in cache = m = 2^r
- Size of tag = (s – r) bits

The blocks of main memory are assigned to lines of the cache as follows :

Cache Line	Main memory blocks assigned
0	0, m, 2m, ……. , 2^s-m
1	1, m+1, 2m+1, ……. 2^s-m+1
.	.
.	.
.	.
m-1	m-1, 2m-1, 3m-1, ……., 2^s-1

Using a part of address as a line number provides a unique mapping of each block of main memory into the cache.

Now, Suppose a block is read from main memory into the cache at position 'p'. To distinguish it from other candidate blocks for that postion 'p', we need to store the tag field (most significant s-r bits) along with the data in the cache.

	Tag	Line	Word
Main memory address =	8	14	2

For our example configuration the mapping becomes as follows :

Cache Line	Starting memory address of block
0	000000, 010000, …….., FF0000
1	000004, 010004, …….., FF0004
.	.
.	.
.	.
2^{14}-1	00FFFC, 01FFFC, …….., FFFFFC

The blocks that map to the same cache line have the same tag. So, at any point of time only one of these blocks can occupy cache. A read operation works as follows: CPU generates a 24-bit address. The 14-bit line number is used as an index to access a particular cache line. Matching logic is applied and if the 8-bit number matches the tag field currently stored in that line, then the remaining 2-bit number is used to select one bytes of the total 4 bytes in that particular line. In case of a miss, the 22-bit (TAG and line field) is used to fetch a block from the memory.

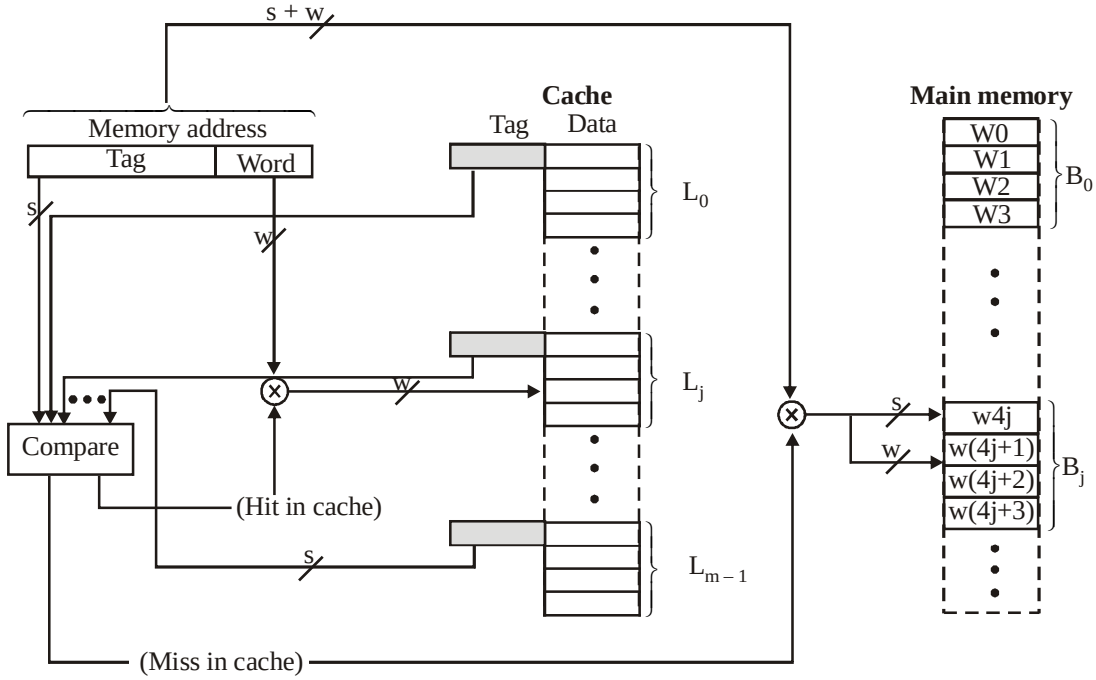
Advantage of Direct mapping is that it is simple and easy to implement. Disadvantage is that "Thrashing" may occur. It refers to the phenomenon when a program repeatedly reference two block from main memory that are candidate blocks for same cache line, so that only one of them can occupy the cache. Now, as the program repeatedly references these two blocks, these will kick-out each other on every memory reference hence significantly decreasing the Hit-Ratio, Even all other cache is empty.

2. Associative Mapping :

This method is used to overcome the problems which are there in Direct mapping. This enables us to store a block of main memory at any random place in the cache. In this method, the address is treated as a TAG and a Word field. No line field is there in this case. The tag field uniquely identifies a block of main memory. In this mapping any block of main memory can occupy any place in the cache, so it is also called **fully associative cache.** The logic is explained in figure:

	Tag	Word
Main memory address =	22	2

When the cache is full and a block need to be stored in cache, the question arises which block in the cache it would replace? Many replacement algorithms have been studied to maximize the hit ratio. Common replacement algorithms are :



Fully Associative Cache Organization

FIFO(First in first out): The block that is oldest in the cache becomes the victim, even if it is still being used.

LRU (Least recently used): The block becomes the victim that was last used at time t, and all other blocks are used atleast once after time t.

MRU(Most recently used) : It is based on the principle that the block that are old in cache will continue to be used, So in this case the latest used block becomes the victim.

The disadvantage of Associative mapping is that implementation is complicated as extra logic circuits are required to examine the tags of all cache lines in parallel.

- Address length = (s + w) bits
- Number of addressable units
  $= 2^s + w$ words or bytes
- Block size = line size = 2^w words or bytes
- Number of blocks in main memory
  $= \frac{2^{s+w}}{2^w} = 2^s$
- Number of lines in set = k
- Number of sets V = 2^x
- Number of lines in cache = kv = k × 2^d
- Size of tag = (s – d) bits

2. Set Associative Mapping :

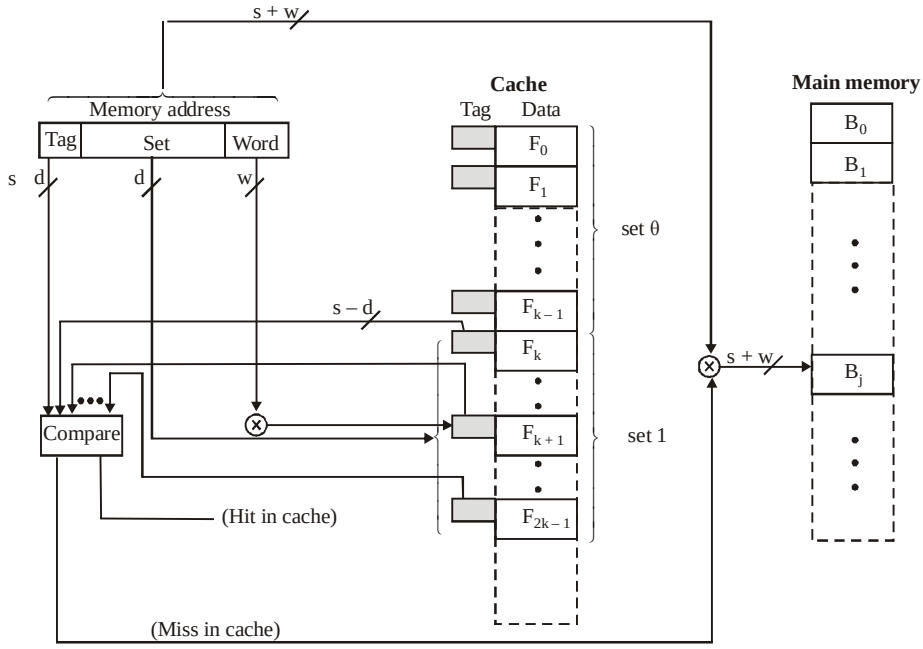
To overcomes the problems arising from above two techniques, Set associative mapping has been developed. This combines the simplicity of direct mapping with the power of the associative mapping. In this case, the cache is divided into v sets, each containing k lines.

m = v x k

I = j modulo v Where, i=cache set number ; j = main memory block number ; m = number of cache lines

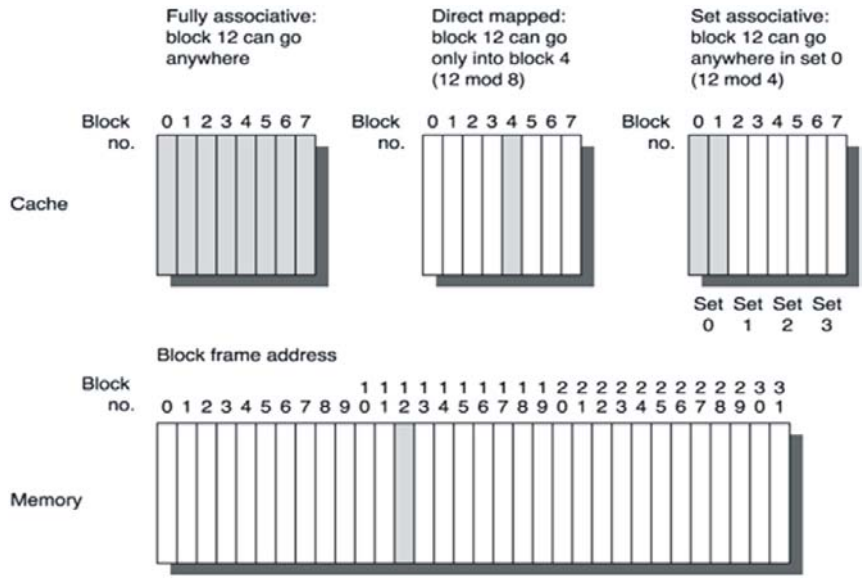
This system is called k-way set associative mapping. In this case, one block B_i of main memory can be mapped to any of the lines of set i. The address is divided into three fields : Tag, Set and Word. The d-bits (Set Bits) specify one of the 2^d sets. The s-bits of the tag and set field specify one of the 2^s blocks of main memory.

The k-way set associative logic is explained in following example :



K-way set associative logic

Three Major Placement Schemes



Number of Tag and Index Bits

Each word in cache has unique index (local addr.)Number of index bits = $\log_2 w$

Index bits are shared with block offset when a block contains more words than 1

Assume partitions of w words each in the main memory.

W/w such partitions, each identified by a tag Number of tag bits = $\log_2(W/w)$

Where Size of main memory =W words and Cache size= w words.

Example 8: A More realistic cache.Consider 4 GB, byte-addressable main memory: 1G words; byte address is 32 bits wide: $b_{31} \ldots b_{16}\ b_{15} \ldots b_2\ b_1\ b_0$. Each word is 32 bits wide

Assume that cache block size is 1 word (32 bits data) and it contains 64 KB data, or 16K words, i.e., 16K blocks.

Sol: Number of cache index bits = 14, because $16K = 2^{14}$

- Tag size = 32 – byte offset – #index bits = 32 – 2 – 14 = 16 bits

Cache requires, for each word:

- 16 bit *tag*, and one *valid bit*
- Total storage needed in cache

= #blocks in cache × (data bits/block + tag size + valid bits)

$= 2^{14}(32+16+1) = 16\times2^{10}\times49 = 784\times2^{10}$ bits = 784 Kb = 98 KB

Physical storage/Data storage = 98/64 = 1.53

Example 9: Consider 4 GB, byte-addressable main memory.1Gwords; byte address is 32 bits wide: b31…b16 b15…b2 b1 b0 .Each word is 32 bits wide.Assume that cache block size is 4 words (128 bits data) and it contains 64 KB data, or 16K words, i.e., 4K blocks.

Number of cache index bits = 12, because $4K = 2^{12}$

Tag size = 32 – byte offset – #block offset bits – #index bits

= 32 – 2 – 2 – 12 = 16 bits

Cache requires, for each word:

- 16 bit *tag*, and one *valid bit*
- Total storage needed in cache

= #blocks in cache × (data bits/block + tag size + valid bit)

$= 2^{12}(4\times32+16+1) = 4\times2^{10}\times145 = 580\times2^{10}$ bits =580 Kb = 72.5 KB

Physical storage/Data storage = 72.5/64 = 1.13

Cache size equation

Simple equation for the size of a cache:

(Cache size) = (Block size) × (Number of sets)

× (Set Associativity)

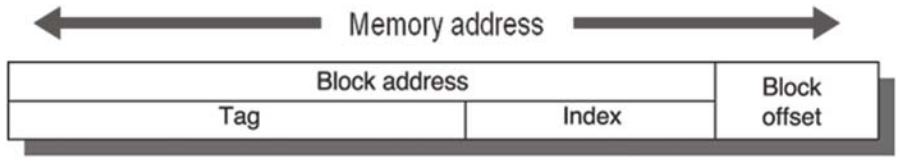
- Can relate to the size of various address fields:

(Block size) = $2^{(\#\text{ of offset bits})}$

(Number of sets) = $2^{(\#\text{ of index bits})}$

(# of tag bits) = (# of memory address bits)

– (# of index bits) – (# of offset bits)



Cache Design

The level's design is described by four behaviors:

Block Placement: Where could a new block be placed in the given level?

Block Identification: How is a existing block found, if it is in the level?

Block Replacement: Which existing block should be replaced, if necessary?

Write Strategy: How are writes to the block handled?

Handling a Miss

Miss occurs when data at the required memory address is not found in cache.

Controller actions:

- If cache is full
  - select the least recently used (LRU) block in cache for over-writing
  - If selected block has inconsistent data, take proper action
- Copy the block containing the requested address from memory

Miss During Instruction Fetch

Send original PC value (PC – 4) to the memory.Instruct main memory to perform a read and wait for the memory to complete the access.Write cache entry.Restart the instruction whose fetch failed.

Writing to Memory

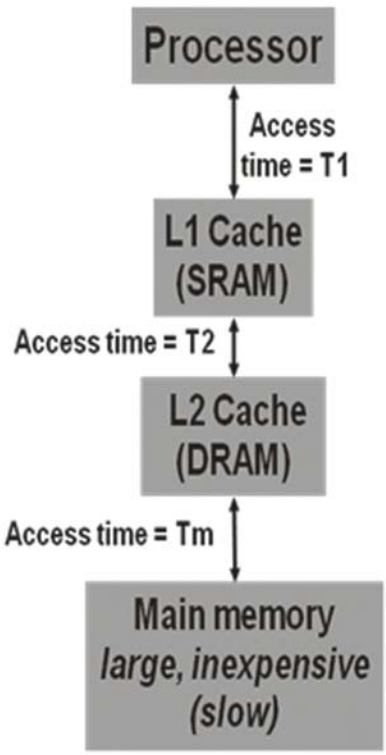
Cache and memory become *inconsistent* when data is written into cache, but not to memory – *the cache coherence problem.*Strategies to handle inconsistent data:

- Write-through
  - Write to memory and cache simultaneously always.
  - Write to memory is ~100 times slower than to (L1) cache.
- Write-back
  - Write to cache and mark block as "dirty".
  - Write to memory occurs later, when dirty block is cast-out from the cache to make room for another block

Writing to Memory: Write-Back

Write-back (or copy back) writes only to cache but sets a "dirty bit" in the block where write is performed.When a block with dirty bit "on" is to be overwritten in the cache, it is first written to the memory."Unnecessary" writes may occur for both write-through and write-back

- write-through has extra writes because each store instruction causes a transaction to memory (e.g. eight 32-bit transactions versus 1 32-byte burst transaction for a cache line)
- write-back has extra writes because unmodified words in a cache line get written even if they haven't been changed
- penalty for write-through is much greater, thus write-back is far more popular



Cache Hierarchy

Average access time

= T1 + (1 – h1) [T2 + (1 – h2)Tm]

Where

- T1 = L1 cache access time (smallest)
- T2 = L2 cache access time (small)
- Tm = memory access time (large)
- h1, h2 = hit rates ($0 \le h1, h2 \le 1$)

Average access time reduces by adding a cache.

Cache Miss Behavior

If the tag bits do not match, then a miss occurs.

Upon a cache miss:

- The CPU is stalled
- Desired block of data is fetched from memory and placed in cache.
- Execution is restarted at the cycle that caused the cache miss.
- Recall that we have two different types of memory accesses:
- reads (loads) or writes (stores).

Thus, overall we can have 4 kinds of cache events:
read hits, read misses, write hits and write misses.

Cache Hits vs. Cache Misses

Consider the write-through strategy: every block written to cache is automatically written to memory.

- Pro: Simple; memory is always up-to-date with the cache
  - No write-back required on block replacement.
- Con: Creates lots of extra traffic on the memory bus.
  - Write hit time may be increased if CPU must wait for bus.

One solution to write time problem is to use a write buffer to store the data while it is waiting to be written to memory.After storing data in cache and write buffer, processor can continue execution. Alternately, a write-back strategy writes data to main memory only a block is replaced.

- Pros: Reduces memory bandwidth used by writes.
- Cons: Complicates multi-processor systems

Hit/Miss Rate, Hit Time, Miss Penalty

The hit rate or hit ratio is fraction of memory accesses found in upper level.
The miss rate (= 1 – hit rate) is fraction of memory accesses not found in upper levels.
The hit time is the time to access the upper level of the memory hierarchy, which includes the time needed to determine whether the access is a hit or miss.The miss penalty is the time needed to replace a block in the upper level with a corresponding block from the lower level (may include the time to write back an evicted block).

Cache Performance Analysis: Performance is always a key issue for caches.

- We consider improving cache performance by:
  (1) reducing the miss rate, and
  (2) reducing the miss penalty.

For (1) we can reduce the probability that different memory blocks will contend for the same cache location.For (2), we can add additional levels to the hierarchy, which is called multilevel caching.

We can determine the CPU time as

$$CPUTime = (CC_{CPU\ Execution} + CC_{MemoryStalls}) \times t_{CC}$$

Cache Performance

The memory-stall clock cycles come from cache misses.It can be defined as the sum of the stall cycles coming from writes + those coming from reads:

Memory-Stall CC = Read-stall cycles + Write-stall cycles, where

$$Read-stall\ cycles = \frac{Reads}{Program} \times Read\ Miss\ Rate \times Read\ Miss\ Penalty$$

$$Write-stall\ cycles = \left(\frac{Writes}{Program} \times Write\ Miss\ Rate \times Write\ Miss\ Penalty \right) + WriteBufferStalls$$

Cache Performance Formulas

Useful formulas for analyzing ISA/cache interactions :

(CPU time) = [(CPU cycles) + (Memory stall cycles)]
× (Clock cycle time)
(Memory stall cycles) = (Instruction count) ×
(Accesses per instruction) × (Miss rate) × (Miss penalty)

You Can split access time into instructions & data:

Avg. mem. acc. time =
(% instruction accesses) × (inst. mem. access time) + (% data accesses)
× (data mem. access time)

Another simple formula:

CPU time = (CPU execution clock cycles + Memory stall clock cycles) ×
cycle time

Useful for exploring ISA changes

Can break stalls into reads and writes:

Memory stall cycles =
(Reads × read miss rate × read miss penalty) + (Writes × write miss rate × write miss penalty)

Factoring out Instruction Count

$$CPU\ time = IC \times Clock\ cycle\ time \times \left(CPI_{exec} + Accesses \times Miss\ rate \times Miss\ penalty\right)$$

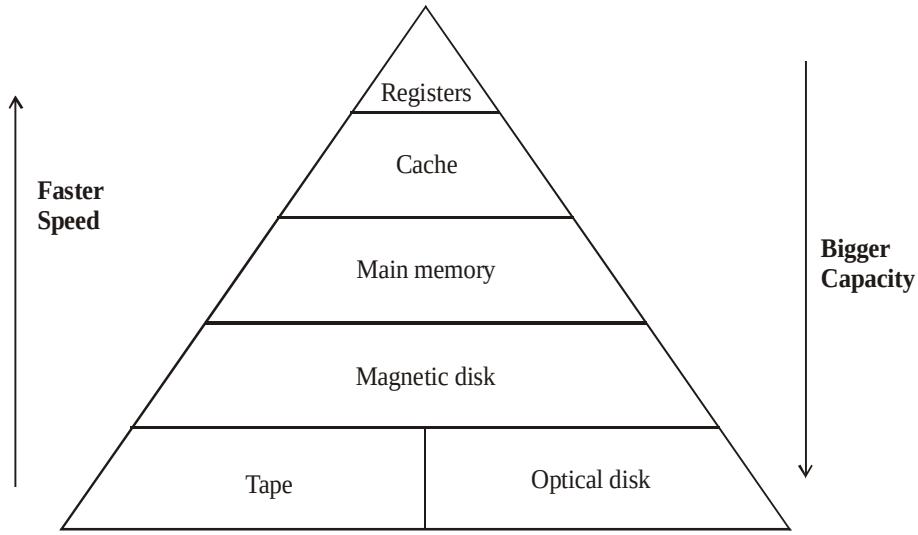
$$\frac{Accesses}{instruction} \times Miss\ rate \rightarrow \frac{Misses}{instruction}$$

SECONDARY MEMORY

The size of the main memory is very small if large data need to be stored in it. Further, the main memory is volatile in nature i.e. the contents are lost when power supply is stopped. To overcome these another memory is used in a computer system called secondary memory. This is large as well as non-volatile in nature.

Memory Hierarchies

Following diagram shows memory hierarchy in a modern computer system :



A five-level memory hierarchy

Among all the CPU registers are fastest that can be accessed at a speed equivalent to the CPU, however they have very small capacity. Next is the cache memory, which is a little slower than registers and size range is from few KB to few MB. Next is Main memory, little more slower and size currently ranging from 16 MB to upto few GB. Then there are magnetic disks substantially slower but with high storage capacity from 200 GB upto few TeraBytes. Finally, we have magnetic tape and optical disks for archival storage. As we move down the hierarchy, three key parameters increase. First, the access time gets bigger. CPU registers can be accessed in a few nanoseconds. Cache memories take a small multiple of CPU registers. Main memory accesses are typically a few tens of nanoseconds. Now comes a big gap, as disk access times are at least 10 msec, and tape or optical disk access can be measured in seconds if the media have to be fetched and inserted into a drive. Second, the storage capacity increases as we go downward. CPU registers are good for perhaps 128 bytes, caches for a few megabytes, main memories for tens to thousands of megabytes, magnetic disks for a few gigabytes to tens of gigabytes. Tapes and optical disks are offline so can be as much as needed.

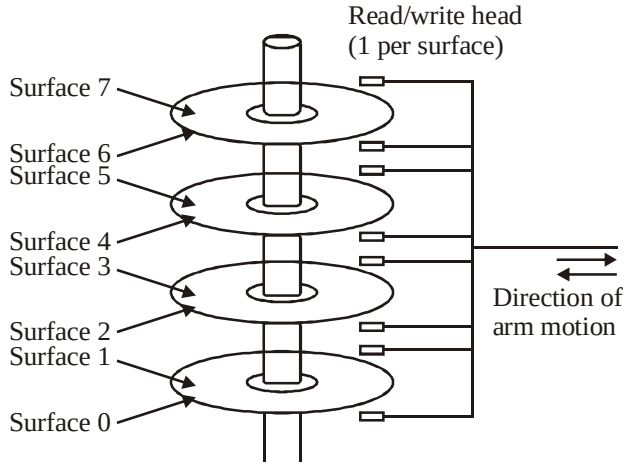
Magnetic Disks

Magnetic disks play two roles in computer systems:

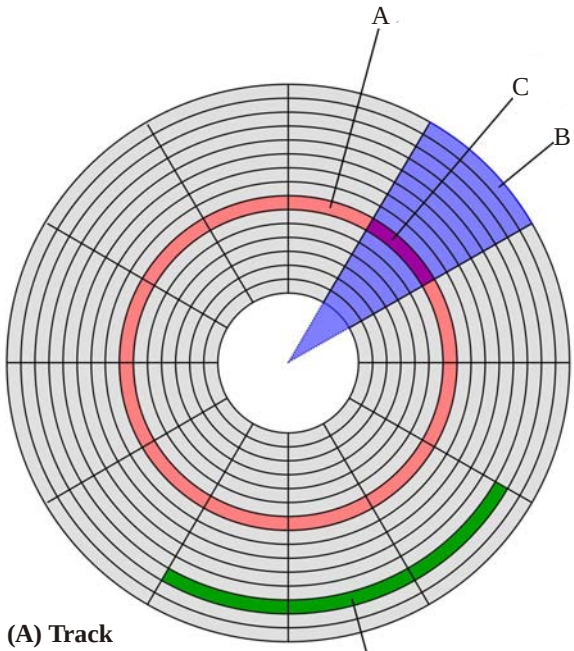
- Long-term, nonvolatile storage for files, even when no programs are running
- A level of the memory hierarchy below main memory used as a backing store for virtual memory during program execution

A magnetic disk consists of a collection of *platters* (generally 1 to 12), rotating on a spindle at 3,600 to 15,000 revolutions per minute (RPM). These platters are metal or glass disks covered with magnetic recording material on both sides, so 10 platters have 20 recording surfaces.

The disk surface is divided into concentric circles, called ***tracks*****.** Each track in turn is divided into ***sectors.*** A sector is the smallest unit that can be read or written.



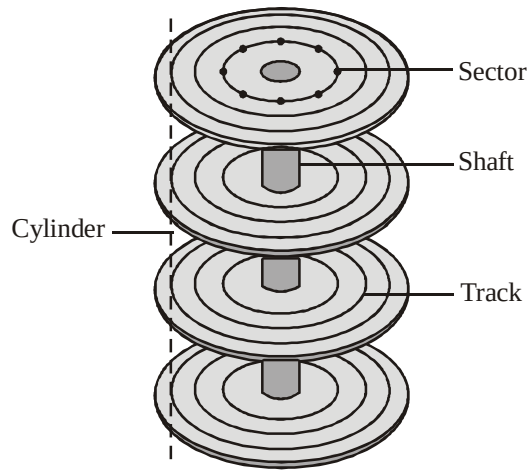
A disk with four platters



(A) Track
(B) Geometrical sector
(C) Track sector
(D) Cluster

Recording more sectors on the outer tracks than on the inner tracks is called ***constant bit density,*** and is mostly practiced now a days. To read and write information into a sector, a movable *arm* containing a *read/write head* is located over each surface. Rather than represent each recorded bit individually, groups of bits are recorded using a run-length-limited code. The arms for all surfaces are connected together and move in conjunction, so that all arms are over the same track of all surfaces. The term *cylinder* is used to refer to all the tracks under the arms at a given point on all surfaces.

To read or write a sector, the disk controller sends a command to move the arm over the proper track. This operation is called a *seek*, and the time to move the arm to the desired track is called *seek time*. The time for the requested sector to rotate under the head is the ***rotational latency*** **or** ***rotational delay***. Note that there are two mechanical components to a disk access. It takes several milliseconds on average for the arm to move over the desired track and several milliseconds on average for the desired sector to rotate under the read/write head. The next component of disk

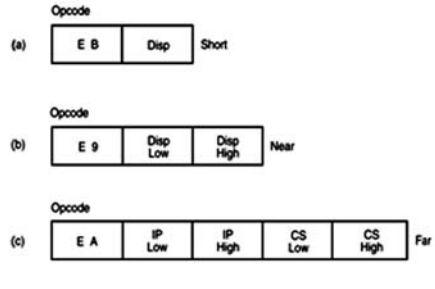


access, ***transfer time***, is the time it takes to transfer a block of bits, typically a sector, under the read-write head. A Disk Controller is associated with each drive that controls the drive. Some controllers contain a full CPU. The controller's tasks include accepting commands from the software, such as READ, WRITE, and FORMAT (writing all the preambles), controlling the arm motion, detecting and correcting errors, and converting 8-bit bytes read from memory into a serial bit stream and vice versa. Some controllers also handle buffering of multiple sectors, caching sectors read for potential future use, and remapping bad sectors. This latter function is caused by the existence of sectors with a bad (permanently magnetized) spot. When the controller discovers a bad sector, it replaces it by one of the spare sectors reserved for this purpose within each cylinder or zone.

Optical Disks

Optical disks are another type of secondary memory. Many types of optical disks are available in the market like CD(Compact disks), DVD (Digital versatile disks) etc. CD-R are write once CDs i.e. data can be written to them only once. CD-RW on the other hand are rewritable CDs i.e. data can be written and erased many times. Similar variations DVD-R and DVD-RW are also available in the market. Optical disks encodes binary data in the form of **"Pits"**: binary value of 0 or off, due to lack of reflection when read and **"lands"**: binary value of 1 or on, due to a reflection when read, on a special material often aluminium on one of its flat surfaces. The data is stored on the disc with a laser or stamping machine.

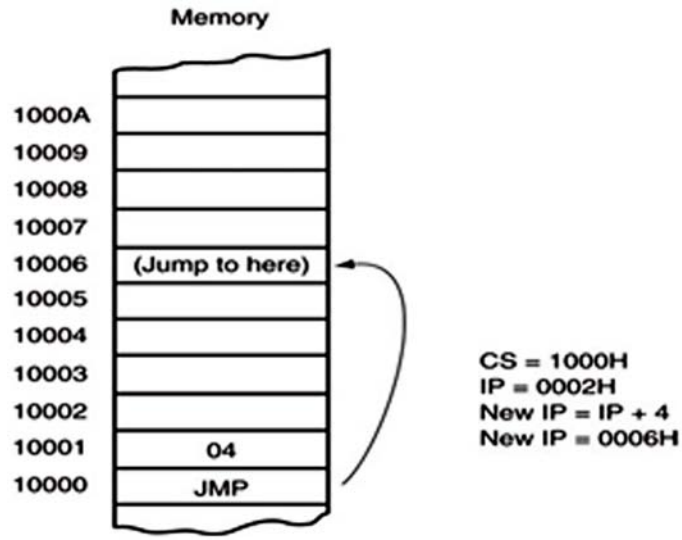
Program Control Instructions

JUMP GROUP		
Allows programmer to skip program sections and branch to any part of memory for the next instruction. - A conditional jump instruction allows decisions based upon numerical tests. - results are held in the flag bits, then tested by conditional jump instructions • LOOP and conditional LOOP are also forms of the jump instruction.		
Unconditional Jump (JMP) Short jump is a 2-byte instruction that allows jumps or branches to memory locations within +127 and – 128 bytes - from the address following the jump The short and near jumps are often called intrasegment jumps.	Near jump 3-byte near jump allows a branch or jump within ±32K bytes from the instruction in the current code segment.	Far jump 5-byte far jump allows a jump to any memory location within the real memory system. Far jumps are called intersegment jumps
		

Short Jump

- Called **relative jumps** because they can be moved, with related software, to any location in the current code segment without a change.
  - jump address is not stored with the opcode
  - a **distance**, or displacement, follows the opcode
- The short jump displacement is a distance represented by a 1-byte signed number whose value ranges between +127 and –128.

A short jump to four memory locations beyond the address of the next instruction is shown below.

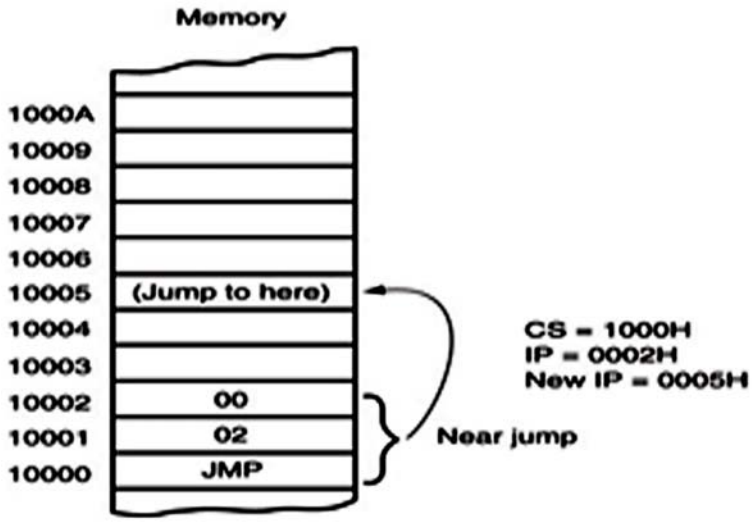


– when the microprocessor executes a short jump, the displacement is sign-extended and added to the instruction pointer (IP/EIP) to generate the jump address within the current code segment.
– The instruction branches to this new address for the next instruction in the program.When a jump references an address, a label normally identifies the address. The JMP NEXT instruction is an example.it jumps to label NEXT for the next instruction very rare to use an actual hexadecimal address with any jump instruction The label NEXT must be followed by a colon (NEXT:) to allow an instruction to reference itif a colon does not follow, you cannot jump to it the only time a colon is used is when the label is used with a jump or call instruction.

Near Jump

A near jump passes control to an instruction in the current code segment located within ±32K bytes from the near jump instruction.

– distance is ±2G in 80386 and above when operated in protected mode
– Near jump is a 3-byte instruction with opcode followed by a signed 16-bit displacement.
  Signed displacement adds to the instruction pointer (IP) to generate the
– jump address. because signed displacement is ±32K, a near jump can jump to any memory location within the current real mode code segment .
– A near jump that adds the displacement (0002H) to the contents of IP.



Far Jump

Obtains a new segment and offset address to accomplish the jump:

– bytes 2 and 3 of this 5-byte instruction contain the new offset address

- bytes 4 and 5 contain the new segment address
- in protected mode, the segment address accesses a descriptor with the base address of the far jump segment
- offset address, either 16 or 32 bits, contains the offset address within the new code segment

A far jump instruction replaces the contents of both CS and IP with 4 bytes following the opcode

- The far jump instruction sometimes appears with the FAR PTR directive.
  - another way to obtain a far jump is to define a label as a far label
  - a label is far only if it is external to the current code segment or procedure
- The JMP UP instruction references a far label.
  - label UP is defined as a far label by the EXTRN UP:FAR directive

Jumps with Register Operands

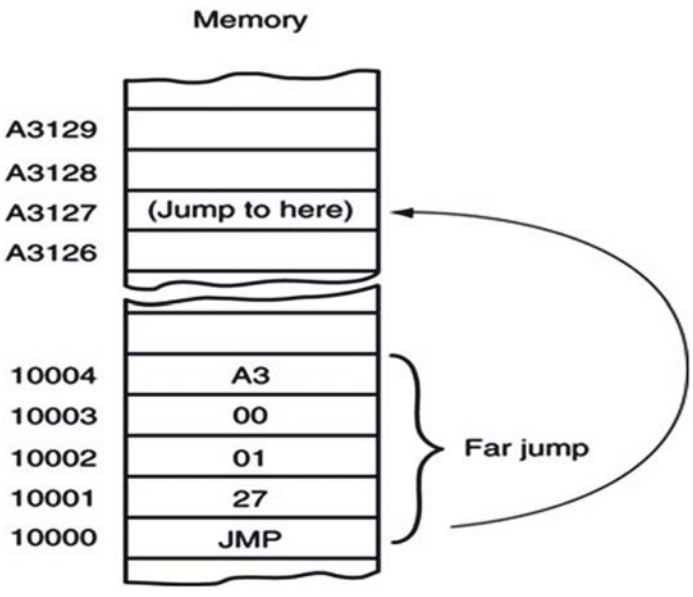
- Jump can also use a 16- or 32-bit register as an operand.
  - automatically sets up as an **indirect jump**
  - address of the jump is in the register specified by the jump instruction
- Unlike displacement associated with the near jump, register contents are transferred directly into the instruction pointer.

An indirect jump does not add to the instruction pointer

- JMP AX, for example, copies the contents of the AX register into the IP.
  - allows a jump to any location within the current code segment

Conditional Jumps and Conditional Sets

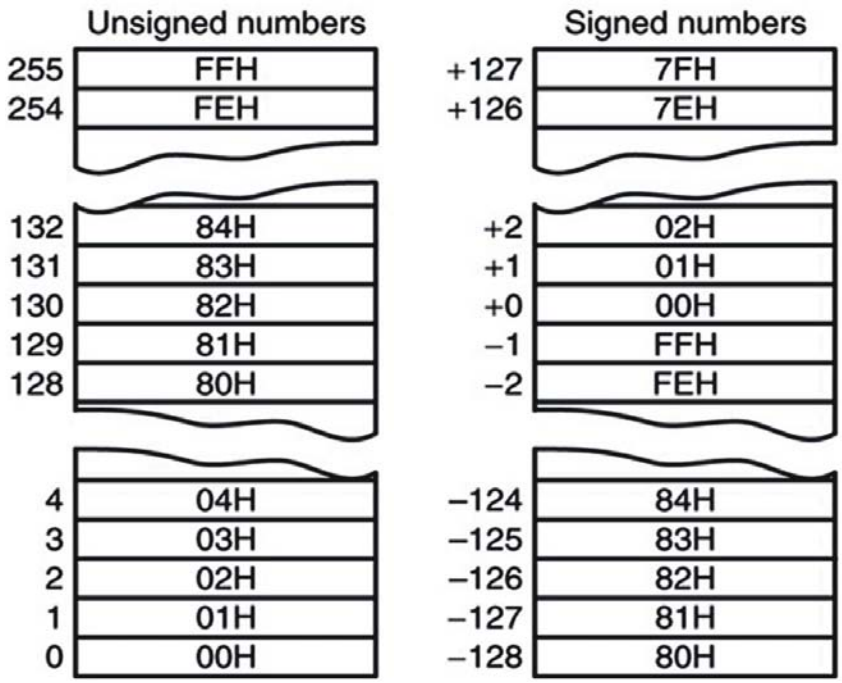
- Always short jumps in 8086 - 80286.
  - limits range to within +127 and –128 bytes from the location following the conditional jump



- In 80386 and above, conditional jumps are either short or near jumps (±32K).
  - in 64-bit mode of the Pentium 4, the near jump distance is ±2G for the conditional jumps
- Allows a conditional jump to any location within the current code segment.
- Conditional jump instructions test flag bits:
  - sign (S), zero (Z), carry (C)
  - parity (P), overflow (0)
- If the condition under test is true, a branch to the label associated with the jump instruction occurs.
  - if false, next sequential step in program executes
  - for example, a JC will jump if the carry bit is set
- Most conditional jump instructions are straightforward as they often test one flag bit.
  - although some test more than one
- Because both signed and unsigned numbers are used in programming.

Because the order of these numbers is different, there are two sets of conditional jump instructions for magnitude comparisons

Signed and unsigned numbers follow different orders



- When signed numbers are compared, use the JG, JL, JGE, JLE, JE, and JNE instructions.
  - terms *greater than* and *less than* refer to signed numbers
- When unsigned numbers are compared, use the JA, JB, JAB, JBE, JE, and JNE instructions.
  - terms *above* and *below* refer to unsigned numbers
- Remaining conditional jumps test individual flag bits, such as overflow and parity.

The Conditional Set Instructions

- 80386 - Core2 processors also contain conditional set instructions.
  - conditions tested by conditional jumps put to work with the conditional set instructions
  - conditional set instructions set a byte to either 01H or clear a byte to 00H, depending on the outcome of the condition under test

Useful where a condition must be tested at a point much later in the program.

LOOP

- A combination of a decrement CX and the JNZ conditional jump.
- In 8086 - 80286 LOOP decrements CX.
  - if CX != 0, it jumps to the address indicated by the label
  - If CX becomes 0, the next sequential instruction executes

In 80386 and above, LOOP decrements either CX or ECX, depending upon instruction mode

- In 16-bit instruction mode, LOOP uses CX; in the 32-bit mode, LOOP uses ECX.
  - default is changed by the LOOPW (using CX) and LOOPD (using ECX) instructions 80386 - Core2
- In 64-bit mode, the loop counter is in RCX.
  - and is 64 bits wide

There is no direct move from segment register to segment register instruction

Conditional LOOPs

- LOOP instruction also has conditional forms: LOOPE and LOOPNE
- LOOPE (**loop while equal**) instruction jumps if CX != 0 while an equal condition exists.
  - will exit loop if the condition is not equal or the CX register decrements to 0
- LOOPNE (**loop while not equal**) jumps if CX != 0 while a not-equal condition exists.
  - will exit loop if the condition is equal or the CX register decrements to 0
- In 80386 - Core2 processors, conditional LOOP can use CX or ECX as the counter.
  - LOOPEW/LOOPED or LOOPNEW/LOOPNED override the instruction mode if needed
- Under 64-bit operation, the loop counter uses RCX and is 64 bits in width
- Alternates exist for LOOPE and LOOPNE.
  - LOOPE same as LOOPZ
  - LOOPNE instruction is the same as LOOPNZ
- In most programs, only the LOOPE and LOOPNE apply.
- Easier to use assembly language statements IF, ELSE, ELSEIF, and ENDIF to control the flow of the program than to use the correct conditional jump statementOther statements developed include REPEAT–UNTIL and WHILE–ENDW.

WHILE Loops

- Used with a condition to begin the loop.
  - the .ENDW statement ends the loop
- The .BREAK and .CONTINUE statements are available for use with the while loop.
  - .BREAK is often followed by .IF to select the break condition as in .BREAK .IF AL == 0DH
  - .CONTINUE can be used to allow a DO–.WHILE loop to continue if a certain condition is met
- The .BREAK and .CONTINUE commands function the same manner in C++.

REPEAT-UNTIL

Loops

- A series of instructions is repeated until some condition occurs.
- The .REPEAT statement defines the start of the loop.
  - end is defined with the .UNTIL statement, which contains a condition
- An .UNTILCXZ instruction uses the LOOP instruction to check CX for a repeat loop.
- – .UNTILCXZ uses the CX register as a counter to repeat a loop a fixed number of times

PROCEDURES

- A procedure is a group of instructions that usually performs one task.
  - subroutine, method, or function is an important part of any system's architecture
- Disadvantage of procedure is time it takes the computer to link to, and return from it.
  - CALL links to the procedure; the RET (**return**) instruction returns from the procedure
- CALL pushes the address of the instruction following the CALL (**return address**) on the stack.
  - the stack stores the return address when a procedure is called during a program

RET instruction removes an address from the stack so the program returns to the instruction following the CALL

CALL

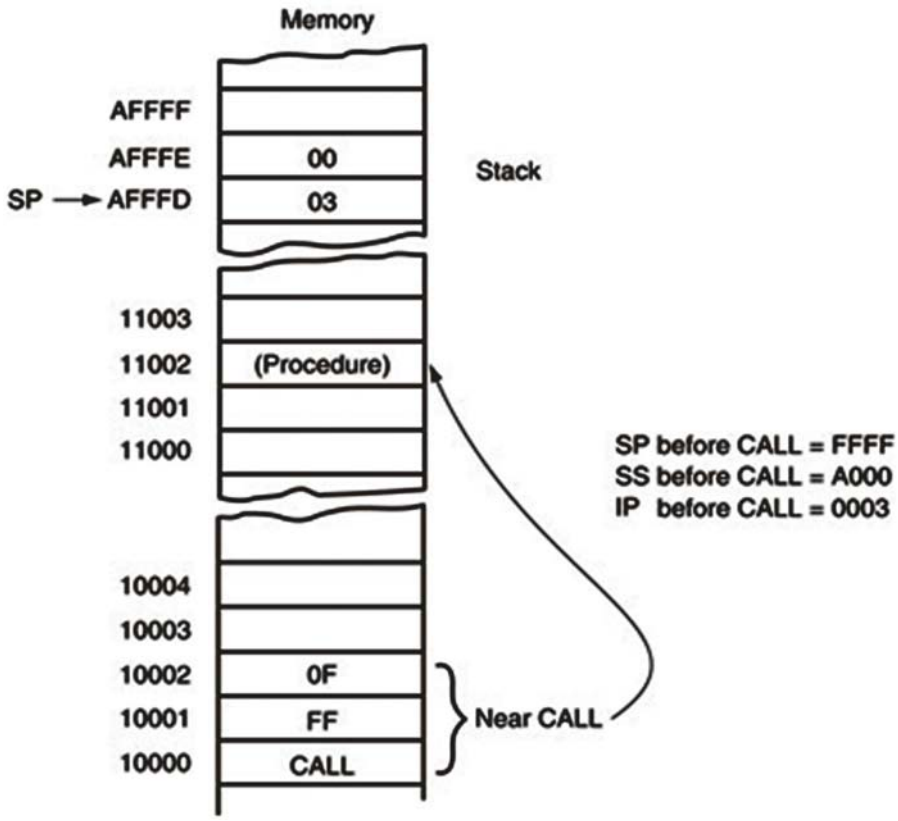
- Transfers the flow of the program to the procedure.
- CALL instruction differs from the jump instruction because a CALL saves a return address on the stack.
- The return address returns control to the instruction that immediately follows the CALL in a program when a RET instruction executes.

Near CALL

- 3 bytes long.
  - the first byte contains the opcode; the second and third bytes contain the displacement
- When the near CALL executes, it first pushes the offset address of the next instruction onto the stack.
  - offset address of the next instruction appears in the instruction pointer (IP or EIP)

  It then adds displacement from bytes 2 & 3 .Kto the IP to transfer control to the procedure

The effect of a near CALL on the stack and the instruction pointer.

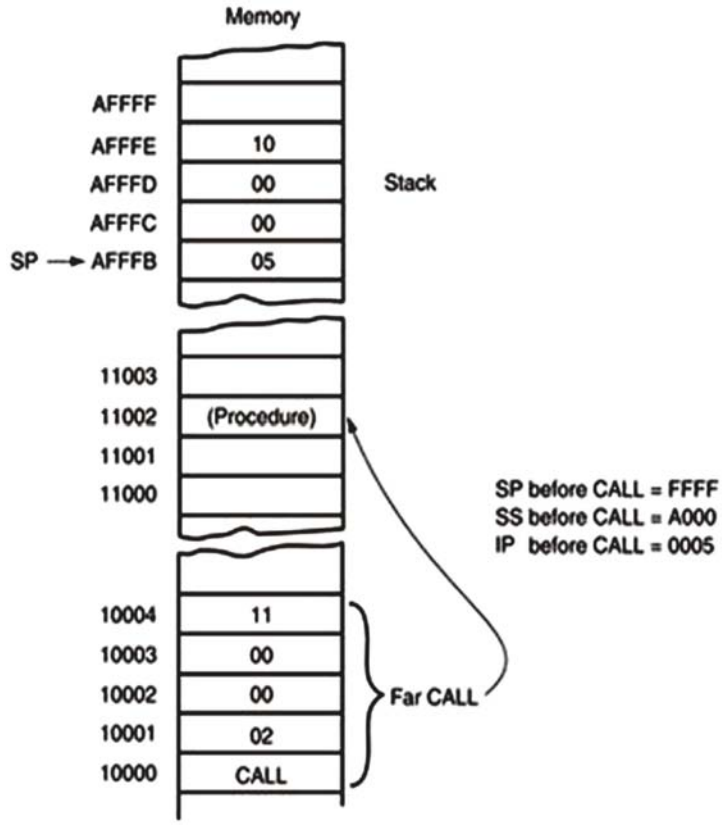


Far CALL

- 5-byte instruction contains an opcode followed by the next value for the IP and CS registers.
  - bytes 2 and 3 contain new contents of the IP
  - bytes 4 and 5 contain the new contents for CS

- Far CALL places the contents of both IP and CS on the stack before jumping to the address indicated by bytes 2 through 5.
- This allows far CALL to call a procedure located anywhere in the memory and return from that procedure.
- Figure below shows how far CALL calls a far procedure.
  - contents of IP and CS are pushed onto the stack
- The program branches to the procedure.
  - A variant of far call exists as CALLF, but should be avoided in favor of defining the type of call instruction with the PROC statement
- In 64-bit mode a far call is to any memory location and information placed onto the stack is an 8-byte number.
  - the far return instruction retrieves an 8-byte return address from the stack and places it into RIP

The effect of a far CALL instruction



CALLs with Register Operands

- An example CALL BX, which pushes the contents of IP onto the stack.
  - then jumps to the offset address, located in register BX, in the current code segment

  Always uses a 16-bit offset address, stored in any 16-bit register except segment registers.

  CALLs with Indirect Memory Addresses
- Particularly useful when different subroutines need to be chosen in a program.
  - selection process is often keyed with a number that addresses a CALL address in a lookup table

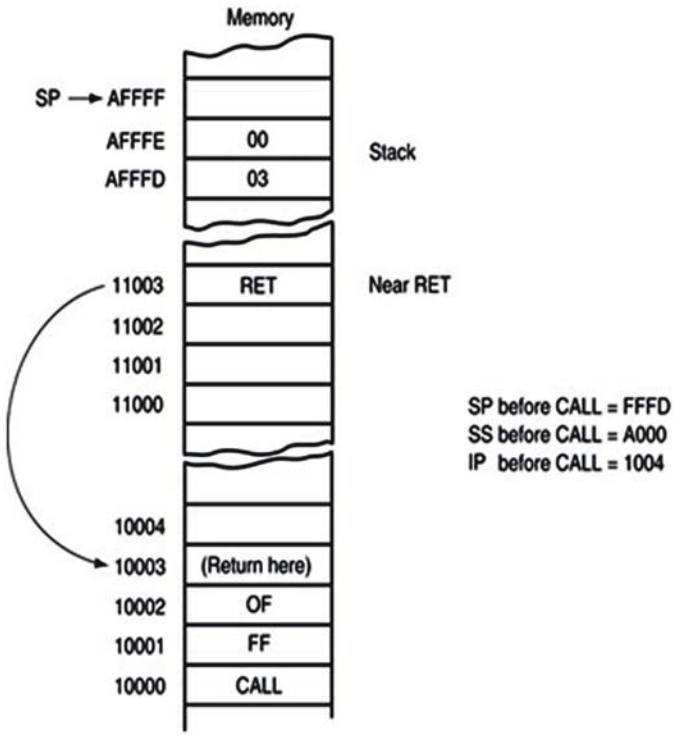
- Essentially the same as the indirect jump that used a lookup table for a jump address.

RET

- Removes a 16-bit number (**near return**) from the stack placing it in IP, or removes a 32-bit number (**far return**) and places it in IP & CS.
  - near and far return instructions in procedure's PROC directive
  - automatically selects the proper return instruction

Figure below shows how the CALL instruction links to a procedure and how RET returns in the 8086–Core2 operating in the real mode

The effect of a near return instruction on the stack and instruction pointer.



INTERRUPTS

- An interrupt is a **hardware-generated CALL**
  - externally derived from a hardware signal
- Or a **software-generated CALL**
  - internally derived from the execution of an instruction or by some other internal event
  - at times an internal interrupt is called an *exception*
- Either type interrupts the program by calling an **interrupt service procedure** (ISP) or interrupt handler.

Interrupt Vectors

- A 4-byte number stored in the first 1024 bytes of memory (00000H–003FFH) in real mode.
  - in protected mode, the vector table is replaced by an interrupt descriptor table that uses 8-byte descriptors to describe each of the interrupts
- 256 different interrupt vectors.

- each vector contains the address of an interrupt service procedure

- Each vector contains a value for IP and CS that forms the address of the interrupt service procedure.
  - the first 2 bytes contain IP; the last 2 bytes CS
- Intel reserves the first 32 interrupt vectors for the present and future products.
  - interrupt vectors (32–255) are available to users
- Some reserved vectors are for errors that occur during the execution of software
  - such as the divide error interrupt
- Three different interrupt instructions available:
  - INT, INTO, and INT 3
- In real mode, each fetches a vector from the vector table, and then calls the procedure stored at the location addressed by the vector.
- In protected mode, each fetches an interrupt descriptor from the interrupt descriptor table.
- Similar to a far CALL instruction because it places the return address (IP/EIP and CS) on the stack.

INTs

- 256 different software interrupt instructions (INTs) available to the programmer.
  - each INT instruction has a numeric operand whose range is 0 to 255 (00H–FFH)
- For example, INT 100 uses interrupt vector 100, which appears at memory address 190H–193H.
  - address of the interrupt vector is determined by multiplying the interrupt type number by 4
- Address of the interrupt vector is determined by multiplying the interrupt type number by 4.
  - INT 10H instruction calls the interrupt service procedure whose address is stored beginning at memory location 40H (10H ´ 4) in the mode
- In protected mode, the interrupt descriptor is located by multiplying the type number by 8
  - because each descriptor is 8 bytes long
- Each INT instruction is 2 bytes long.
  - the first byte contains the opcode
  - the second byte contains the vector type number

When a software interrupt executes, it:
- pushes the flags onto the stack
- clears the T and I flag bits
- pushes CS onto the stack
- fetches the new value for CS from the interrupt vector
- pushes IP/EIP onto the stack
- fetches the new value for IP/EIP from the vector
- jumps to the new location addressed by CS and IP/EIP

- INT performs as a far CALL
  - not only pushes CS & IP onto the stack, also pushes the flags onto the stack
- The INT instruction performs the operation of a PUSHF, followed by a far CALL instruction.
- Software interrupts are most commonly used to call system procedures because the address of the function need not be known.

The interrupts often control printers, video displays, and disk drives

- INT replaces a far CALL that would otherwise be used to call a system function.
  - INT instruction is 2 bytes long, whereas the far CALL is 5 bytes long
- Each time that the INT instruction replaces a far CALL, it saves 3 bytes of memory.
- This can amount to a sizable saving if INT often appears in a program, as it does for system calls.

INT 3

- A special software interrupt designed to function as a breakpoint.
  - a 1-byte instruction, while others are 2-byte
- Common to insert an INT 3 in software to interrupt or break the flow of the software.
  - function is called a breakpoint
  - breakpoints help to debug faulty software
- A breakpoint occurs for any software interrupt, but because INT 3 is 1 byte long, it is easier to use for this function.

INTO

- Interrupt on overflow (INTO) is a conditional software interrupt that tests overflow flag (O).
  - If O = 0, INTO performs no operation
  - if O = 1 and an INTO executes, an interrupt occurs via vector type number 4
- The INTO instruction appears in software that adds or subtracts signed binary numbers.
  - eith these operations, it is possible to have an overflow
- JO or INTO instructions detect the overflow.

Interrupt Control

- Two instructions control the INTR pin.
- The **set interrupt flag** instruction (STI) places 1 in the I flag bit.
  - which enables the INTR pin
- The **clear interrupt flag** instruction (CLI) places a 0 into the I flag bit.
  - which disables the INTR pin

The STI instruction enables INTR and the CLI instruction disables INTR

- In software interrupt service procedure, hardware interrupts are enabled as one of the first steps.
  - accomplished by the STI instruction
- Interrupts are enabled early because just about all of the I/O devices in the personal computer are interrupt-processed.
  - if interrupts are disabled

Controlling the Carry Flag Bit

- The carry flag (C) propagates the carry or borrow in multiple-word/doubleword addition and subtraction.
  - can indicate errors in assembly language procedures
- Three instructions control the contents of the carry flag:
  - STC (set carry), CLC (clear carry), and CMC (complement carry)

Instruction	Function
WAIT	If the WAIT instruction executes while the *BUSY* pin = 1, nothing happens and the next instruction executes. – pin inputs a busy condition when at a logic 0 level – if *BUSY* pin = 0 the microprocessor waits for the pin to return to a logic 1
HLT	Stops the execution of software. There are three ways to exit a halt: – by interrupt; a hardware reset, or DMA operation Often synchronizes external hardware interrupts with the software system
NOP	NOP, which performs absolutely no operation When the microprocessor encounters a NOP, it takes a short time to execute.
ENTER and LEAVE	Used with stack frames, mechanisms used to pass parameters to a procedure through the stack memory. Stack frame also holds local memory variables for the procedure. Stack frames provide dynamic areas of memory for procedures in multiuser environments. • ENTER creates a stack frame by pushing BP onto the stack and then loading BP with the uppermost address of the stack frame. – allows stack frame variables to be accessed through the BP register • ENTER contains two operands: – first operand specifies the number of bytes to reserve for variables on the stack frame – the second specifies the level of the procedure LEAVE instruction removes the stack frame from the stack. The BP register addresses stack frame data.
BOUND	instruction compares the contents of any 16-bit register against the contents of two words of memory: an upper and a lower boundary.

Past GATE Questions Exercise

1. A 5 stage pipelined CPU has the following sequence of stages
IF : Instruction fetch from instruction memory
RD : Instruction decode and register read
EX : Execute: ALU operations for data and address computation
MA : Data memory access : for write access, the register read at RD stage is used
WB : Register write back
Consider the following sequence of instructions:
I_1 : L R_0 loc 1; R_0 < = M [loc_1]
I_2 : A R_0 ; R_0; R_0 < = R_0 + R_0
I_3 : S R_2 ; R_0; R_2 < = R_2 – R_0
Let each state takes on clock cycle.
What is the number of clock cycles taken to complete the above sequence of instructions starting from the fetch of I_1? **[2005, 1 mark]**
(a) 8 (b) 10
(c) 12 (d) 15

2. Consider a direct mapped cache of size 32 kbyte with block size 32 byte. The CPU generates 32 bit address. The number of bits needed for cache indexing and the number of tag bits are respectively **[2005, 1 mark]**
(a) 10, 17 (b) 10, 22
(c) 15, 17 (d) 5, 17

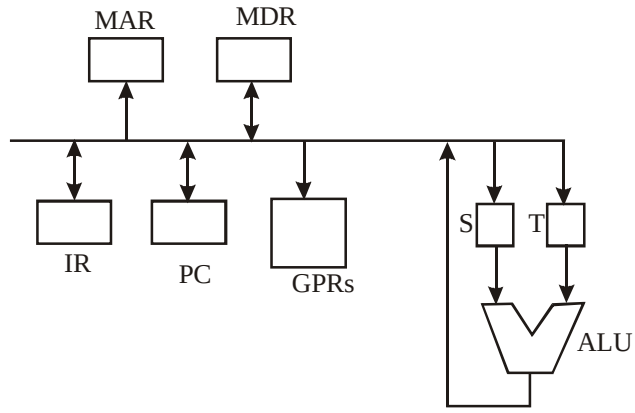
3. Match each of the high level language statements given on the left hand side with the most natural addressing mode from those listed on the right hand side. **[2005, 1 mark]**

List - 1	List - II
P. A[l] = B[l]	1. Indirect addressing
Q. while (* A ++);	2. Indexed addressing
R. int temp = * x;	3. Auto increment

(a) P - 3, Q - 2, R - 1 (b) P - 1, Q - 2, R - 3
(c) P - 2, Q - 4, R - 1 (d) P - 1, Q - 2, R - 3

4. Consider a three word machine instruction:
ADD A[R_0], @ B
The first operand (destination) "A[R_0]" uses indexed addressing mode with R_0 as the index register. The second operand (source) "@B" uses indirect addressing mode. A and B are memory addresses residing at the second and the third words, respectively. The first word of the instruction specifies the opcode, the index register designation and the source and destination addressing modes. During execution of ADD instruction, the two operands are added and stored in the destination (first operand).
The number of memory cycles needed during the execution cycle of the instruction is **[2005, 1 mark]**
(a) 3 (b) 4
(c) 5 (d) 6

Statements for Linked Answer Questions 5 and 6:
Consider the following data path of a CPU:



The ALU, the bus and all the registers in the data path are of identical size. All operations including incrementation of the PC and the GPRs are to be carried out in the ALU. Two clock cycles are needed for memory bus into the MDR.

5. The instruction "ADD R_0, R_1" has the register transfer interpretation R_0 < = R_0 + R_1. The minimum number of clock cycles needed for execution cycle of this instruction is **[2005, 1 mark]**
(a) 2 (b) 3
(c) 4 (d) 5

6. The instruction "CALL R_n, sub" is a two word instruction. Assuming that PC is incremented during the fetch cycle of the first word of the instruction, its register transfer interpretation is
R_n < = PC + 1
PC < = M [PC]
The minimum number of CPU clock cycles needed during the execution cycle of this instruction is **[2005, 1 mark]**
(a) 2 (b) 3
(c) 4 (d) 5

7. Increasing the RAM of a computer typically improved performance because **[2005, 1 mark]**
(a) virtual memory increases
(b) larger RAM are faster
(c) fewer page faults occur
(d) fewer segmentation faults occur

8. What is the swap space in the disk used for?
(a) Saving temporary HTML pages **[2005, 1 mark]**
(b) Saving process data
(c) Storing the super-block
(d) Storing device drivers

9. Normally user programs are prevented from handling I/O directly by I/O instructions in them. For CPUs having explicit I/O instructions, such I/O protection is ensured by having the I/O instructions privileged. In a CPU with memory mapped I/O, there is no explicit I/O instruction. Which one of the following is true for a CPU with memory mapped I/O? **[2005, 1 mark]**

(a) I/O protection is ensured by operating system routine(s)
(b) I/O protection is ensured by a hardware trap
(c) I/O protection is ensured during system configuration
(d) I/O protection is not possible

10. Which one of the following is true for a CPU having a single interrupt request line and a single interrupt grant line? **[2005, 1 mark]**

(a) Neither vectored interrupt nor multiple interrupting devices are possible
(b) Vectored interrupt are not possible but multiple interrupting devices are possible
(c) Vectored interrupt and multiple interrupting are both possible
(d) Vectored interrupt is possible but multiple interrupting devices are not possible

11. Consider a new instruction named branch-on-bit-set (mnemonic bbs). The instruction "bbs reg, pos, label" jumps to label if bit in position pos of register operand reg is one. A register is 32 bit wide and the bits are numbered 0 to 32, bit in position 0 being the least significant. Consider the following emulation of this instruction on a processor that does not have bbs implemented.

tem ← reg and mask
Branch to lable if temp is non-zero

The variable temp is a temporary register. For correct emulation, the variable mask must be generated by **[2006, 2 marks]**

(a) mask ← 0 × 1 << pos
(b) mask ← 0 × ffffffff << pos
(c) mask ← pos
(d) mask ← 0 × f

12. A CPU has a five-stage pipeline and runs at 1 GHz frequency. Instruction fetch happens in the first stage of the pipeline. A conditional branch instruction computes the target addresses and evaluates the condition in the third stage of the pipeline. The processor stop fetching new instructions following a conditional branch until the branch outcome is known. A program executes 10^9 instructions out of which 20% are conditional branches. If each instruction takes one cycle to complete on average, the total execution time of the program is **[2006, 2 marks]**

(a) 1.0 s (b) 1.2 s
(c) 1.4 s (d) 1.6 s

13. A CPU has a cache with block size 64 byte. The main memory has k banks, each bank being c byte wide. Consecutive c-byte chunks are mapped on consecutive banks with wrap-around. All the k blanks can be accessed in parallel, but two accesses to the same bank must be serialized. A cache block access may involve multiple iterations of parallel bank accesses depending on the amount of data obtained by accessing all the k banks is parallel. Each iteration requires decoding the bank numbers to be accessed in parallel and this takes $\frac{k}{2}$ ns. The latency of one bank access is 80 ns. If c = 2 and k = 24, the latecy of retrieving a cache block starting at address zero from main memory is **[2006, 2 marks]**

(a) 92 ns (b) 104 ns
(c) 172 ns (d) 184 ns

Statements for Linked Answer Questions 14 and 15 :

Consider a machine with a byte addressable main memory of 2^{16} byte. Assume that a direct mapped data cache consisting of 32 lines of 64 byte each is used in the system. A 50 × 50 two-dimensional array of bytes is stored in the main memory starting from memory location 1100H. Assume that the data cache is initially empty. The complete array is accessed twice. Assume that the contents of the data cache do not change in between the two accesses.

14. How many data cache misses will occur in total? **[2007, 2 marks]**

(a) 48 (b) 50
(c) 56 (d) 59

15. Which of the following lines of the data cache will be replaced by new blocks in accessing the array for the second time? **[2007, 2 marks]**

(a) line 4 to line 11 (b) line 4 to line 12
(c) line 0 to line 7 (d) line 4 to line 8

Consider Data for Questions 16, 17 and 18

Consider the following program segments. Here R_1, R_2 and R_3 are purpose registers.

Instruction	Operation	Instruction size (number of words)
MOV R_1, 3000	$R_1 \leftarrow M[3000]$	2
LOOP MOV R_2, R_3	$R_2 \leftarrow M[R_3]$	1
ADD R_2,	$R_2 \leftarrow R_1 + R_2$	1
MOV R_3, R_2	$M[R_3] \leftarrow R_2$	1
INC R_3	$R_3 \leftarrow R_3 + 1$	1
DEC R_1	$R_1 \leftarrow R_1 - 1$	1
BNZ LOOP	Branch on not zero	2
HALT	Stop	1

Assume that the content of memory location 3000 is 10 and the content of the register R_3 is 2000. The content of each of the memory locations from 2000 to 2010 is 100. The program is loaded from the memory location 1000. All the numbers are in decimal.

16. Assume that the memory is word addressable. After the execution of this program, the content of memory location 2010 is **[2007, 2 marks]**

(a) 10 (b) 11
(c) 20 (d) 21

17. Assume that the memory is word addressable. After the execution of this program, the content of memory location 2010 is **[2007, 2 marks]**

(a) 100 (b) 101
(c) 102 (d) 110

18. Assume that the memory is byte addressable and the word size is 32 bit. If an interrupt occurs during the execution of the instruction INC R_3, what return address will be pushed on to the stack? **[2007, 2 marks]**

(a) 1005 (b) 1020
(c) 1024 (d) 1040

19. Consider a pipelined processor with the following four stages

IF : Instruction Fetch
ID : Instruction Decode and Operand Fetch
EX : Execute
WB : Write Black

The IF, ID and WB stages take 1 clock cycle each to complete the operation.

- The number of clock cycle for the EX stage depends on the instruction. the ADD and SUB instructions need 1 clock cycle and the MUL instruction needs 3 clock cycles in the EX stage. Operand forwarding is used in the pipelined processor. What is the number of clock cycles taken to complete the following sequence of instructions:

[2007, 2 marks]

ADD	R_2, R_1, R_0	$R_2 \leftarrow R_1 + R_0$
MUL	R_4, R_3, R_2	$R_4 \leftarrow R_3 * R_2$
SUB	R_6, R_5, R_4	$R_6 \leftarrow R_5 - R_4$

(a) 7 (b) 6
(c) 10 (d) 14

20. A CPU has 24-bit instructions. A program starts at address 300 (in decimal). Which one of the following is a legal program counter (all values in decimal)? **[2007, 1 mark]**

(a) 400 (b) 500
(c) 600 (d) 700

21. Consider a disk pack with 16 surfaces, 128 tracks per surface and 256 sectors per track. 512 byte of data are stored in a bit serial manner in a second. The capacity of the disk pack and the number of bits required to specify a particular section in the disk are respectively **[2007, 1 mark]**

(a) 256 Mbyte, 19 bit (b) 256 Mbyte, 28 bit
(c) 512 Mbyte, 20 bit (b) 64 Gbyte, 28 bit

22. Consider a 4-way set associate cache consisting of 128 lines with a line size of 64 words. The CPU generates a 20-bit address of word in main memory. The number of bits in the TAG, LINE and WORD fields are respectively.

[2007, 1 mark]

(a) 9, 6, 5 (b) 7, 7, 6
(c) 7, 5, 8 (d) 9, 5, 6

Statement for Linked Answer Questions 23 and 24 :

Delayed branching can help in the handling of control hazards.

23. For all delayed conditional branch instructions, irrespective of whether the condition evaluations to true or false. **[2008, 2 marks]**

(a) the instruction following the conditional branch instruction in memory is executed
(b) the first instruction in the fall through path is executed
(c) the first instruction in the taken path is executed
(d) the branch taken longer to execute than any other instruction

24. The following code is to run on a pipelined processor with one branch delay slot: **[2008, 2 marks]**

l_1: ADD $R_2 < R_7 + R_8$
l_2: SUB $R_4 \leftarrow R_5 - R_6$
l_3: ADD $R_1 \leftarrow R_2 + R_3$
l_4: STORE Memory $[R_4] \leftarrow R_1$
BRANCH to Label if $R_1 = 0$

Which of the instructions I_1, I_2, I_3 or I_4 can legitimately occupy the delay slot without any other program modification?

(a) I_1 (b) I_2
(c) I_3 (d) I_4

Common Data for Questions 25, 26 and 27 :

Consider a machine with a 2-way set associative data cache of size 64 kbyte and block size 16 byte. The cache is managed using 32 bit virtual addresses and the page size is 4 kbyte. A program to be run on this machine begins as follows.

```
double ARR [1024] [1024]
    int i, j;
    /* Initialize array ARR to 0.0 */
    for (i = 0; i < 1024; i ++)
    for (j = 0; j < 1024; j ++)
    ARR [i] [j] = 0.0;
```

The size of double is 8 byte. Array ARR is located in memory starting at the beginning of virtual page 0 × FF000 and stored in row major order. The cache is initially empty and no pre-fetching is done. The only data memory references made by the program are those to array ARR.

25. The total size of the tags in the cache directory is **[2008, 2 marks]**

(a) 32 kbit (b) 34 kbit
(c) 64 kbit (d) 68 kbit

26. Which of the following array elements has the same cache index as ARR [0] [0] ? **[2008, 2 marks]**

(a) ARR [0] [4] (b) ARR [4] [0]
(c) ARR [0] [5] (d) ARR [5] [0]

27. The cache hit ratio for this initialization loop is **[2008, 2 marks]**

(a) 0% (b) 25%
(c) 50% (d) 75%

28. Which of the following must be true for the REF (Return From Exception) **[2008, 2 marks]**

1. It must be a TRAP instruction.
2. It must be a privileged instruction.
3. An exception cannot be allowed to occur during execution of an REE instruction.

(a) 1 only (b) 2 only
(c) 1 and 2 (d) 1, 2 and 3

29. Which of the following is/are true of the auto-increment addressing mode? **[2008, 2 marks]**

1. It is used in creating self-relocating code.
2. If it is included in an Instruction Set Architecture, then an additional ALU is required for effective address calculation.
3. The amount of increment depends on the size of the data item accessed.

(a) 1 only (b) 2 only
(c) 3 only (d) 2 and 3

30. For a magnetic disk with concentric circular tracks, the seek latency is not linearly proportional to the seek distance due to **[2008, 1 mark]**

(a) non-uniform distribution of requests
(b) arm starting and stopping inertia
(c) higher capacity of tracks on the periphery of the platter
(d) use of unfair arm scheduling policies

31. A CPU generally handles an interrupt by executing an interrupt service routine **[2009, 1 mark]**

(a) as soon as an interrupt as raised
(b) by checking the interrupt register at the end of fetch cycle
(c) by checking the interrupt register after finishing the execution of the current instruction
(d) by checking the interrupt register at fixed time intervals.

32. How many 32 k × 1 RAM chips are needed to provide a memory capacity of 256 kbyte? **[2009, 1 mark]**

(a) 8 (b) 32
(c) 64 (d) 128

Common Data for Questions 33 and 34 :

A hard disk has 63 sectors per track, 10 platters each with 2 recording surfaces and 1000 cylinders. The address of a sector is given as a triple (c, h, s), where c is the cylinder number, h is the surface number and s is the sector number. Thus, the 0th sector is addressed as $\langle 0,0,0 \rangle$, the 1st sector as $\langle 0,0,1 \rangle$, and so on.

33. The address < 400, 16, 29 > corresponds to sector number **[2010, 2 marks]**

(a) 505035 (b) 505036
(c) 505037 (d) 505038

34. The address of the 1038th sector is **[2010, 2 marks]**

(a) < 0, 15, 31> (b) < 0, 16, 30>
(c) < 0, 16, 31> (d) < 0, 17, 31>

35. Consider a 4-way set associative cache (initially empty) with total 16 cache blocks. The main memory consists of 256 blocks and the request for memory block is in the following order:

0, 255, 1, 4, 3, 8, 133, 159, 216, 129, 63, 8, 48, 32, 73, 92, 155

Which one of the following memory blocks will not be in cache if LRU replacement policy is used? **[2010, 2 marks]**

(a) 3 (b) 8
(c) 129 (d) 216

36. Consider a 4 stage pipeline processor. The number of cycles needed by the four instructions I_1, I_2, I_3, I_4 in stages S_1, S_2, S_3, S_4 is shown below.

	S_1	S_2	S_3	S_4
I_1	2	1	1	1
I_2	1	3	2	2
I_3	2	1	1	3
I_4	1	2	2	2

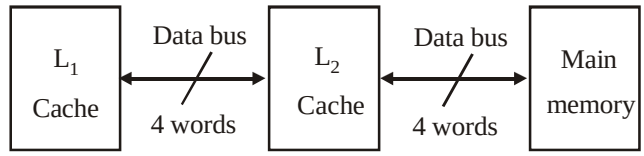
What is the number of cycles needed to execute the following loop? **[2010, 2 marks]**

For (i = 1 to 2) {I_1; I_2; I_3; I_4;}

(a) 16 (b) 23
(c) 28 (d) 30

Statements for Linked Answer Questions 37 and 38 :

The computer system has an L_1 L_2 cache, an L_2 cache and a main memory unit connected as shown below. The block size in L_1 cache is 4 words. The block size in L_2 cache is 16 words. The memory access times are 2 ns, 20 ns and 200 ns, for L_1 cache, L_2 cache and main memory unit respectively.



37. When there is a miss in L_1 cache and a hit in L_2 cache, a block is transferred from L_2 cache to L_1 cache. What is the time taken for this transfer? **[2010, 2 marks]**

(a) 2 ns (b) 20 ns
(c) 22 ns (d) 88 ns

38. When there is a miss in both L_1 cache and L_2 cache, first a block is transferred from main memory to L_2 cache, and then a block is transferred from L_2 cache to L_1 cache. What is the total time taken for these transfer? **[2010, 2 marks]**

(a) 222 ns (b) 888 ns
(c) 902 ns (d) 968 ns

39. A 5-stage pipelined processor has Instruction Fetch (IF), Instruction Decode (ID), Operand Fetch (OF), Perform Operation (PO) and Write Operand (WO) stages. The IF, ID, OF and WO stages take 1 clock cycle each for any instruction. The PO stage takes 1 clock cycle for ADD and SUB instructions, 3 clock cycles for MUL instruction, and 6 clock cycles for DIV instruction respectively. Operand forwarding is used in the pipeline. What is the number of clock cycles needed to execute the following sequence of the instructions? **[2010, 2 marks]**

Instruction	Meaning of instruction
I_0: MUL R_2, R_0, R_1	$R_2 \rightarrow R_0 * R_1$
I_1: DIV R_5, R_3, R_4	$R_5 \rightarrow R_3 / R_4$
I_2: ADD R_2, R_5, R_2	$R_2 \rightarrow R_5 + R_2$
I_3: SUB R_5, R_2, R_6	$R_5 \rightarrow R_2 - R_6$

(a) 13 (b) 15
(c) 17 (d) 19

40. A main memory unit with a capacity of 4 megabyte is built using 1 M × 1 bit DRAM chips. Each DRAM chip has 1 rows of cells with 1 k cells in each row. The time taken for a single refresh operation is 100 ns. The time required to perform one refresh operation on all the cells in the memory unit is **[2010, 1 mark]**

(a) 100 ns (b) $100 * 2^{10}$ ns
(c) $100 * 2^{20}$ ns (d) $3200 * 2^{20}$ ns

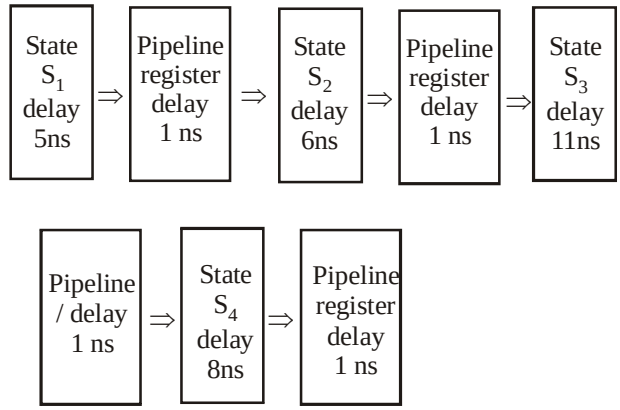
41. On a non-pipelined sequential processor, a program segment, which is a part of the interrupt service routine, is give to transfer 500 byte from an I/O device to memory
Initialize the address register
Initialize the count to 500
LOOP: Load a byte from device
Store in memory at address given by address register
Increment the address register
Decrement the count
If count! = 0 goto LOOP
Assume that each statement in this program is equivalent to a machine instruction which takes one clock cycle to executive if it is a non-load/store instruction. The load-store instructions take two clock cycles to execute. The designer of the system also has an alternate approach of using the DMA controller to implement the same transfer. The DMA controller required 20 clock cycles for initiallization and other overheads. Each DMA transfer cycle takes two clock cycles to transfer one byte of data from the device to the memory. What is the approximate speed up when the DMA controller based design is used in place of the interrupt driven program based input-output? **[2011, 2 marks]**

(a) 3.4 (b) 3.5
(c) 3.3 (d) None of these

42. An 8 kbyte direct mapped write-back cached is organized as multiple blocks, each of size 32 byte. The processor generates 32-bit addresses. The cache controller maintains the tag information for each cache block comprising of the following–
1 Valid bit
1 Modified bit
As many bits as the minimum needed to identify the memory block mapped in the cache.
What is the total size of memory needed at the cache controller to store metadata (tags) for the cache? **[2011, 2 marks]**

(a) 4864 bit (b) 6144 bit
(c) 6656 bit (d) 5376 bit

43. Consider an instruction pipeline with four stages (S_1, S_2, S_3 and S_4) each with combinational circuit only. The pipeline registers are required between each stage and at the end of the last stage. Delays for the stages and for the pipeline registers are as given in the figure **[2011, 2 marks]**



What is the approximate speed up of the pipeline in steady state under ideal conditions when compared to the corresponding non-pipeline implementation?

(a) 4.0 (b) 2.5
(c) 1.1 (d) 3.0

44. A computer handles several interrupt sources of which of the following are relevant? **[2012, 1 mark]**
Interrupt from CPU temperature sensor
Interrupt from Mouse
Interrupt from Keyboard
Interrupt from Hard disk

(a) Interrupt from Hard disk
(b) Interrupt from Mouse
(c) Interrupt from Keyboard
(d) Interrupt from CPU temperature sensor

45. Consider a hypothetical processor with an instruction of type LW (R_1), 20 (R_2). which during execution reads a 32-bit word from memory and stores it in a 32-bit register R1. The effective address of the memory location is obtained by the addition of constant 20 and the contents of register R_2. Which of the following best reflects the addressing mode implemented by this instrument for the operand in memory?

(a) Immediate addressing **[2012, 1 mark]**
(b) Register addressing
(c) Register indirect scalled addressing
(d) Base indexed addressing

46. Let the page fault service time be 10 Ms in a computer with average memory access time being 20 ns. If one page fault is generated for every 10^6 memory accesses, what is the effective access time for the memory? **[2012, 1 mark]**

(a) 21 ns (b) 30 ns
(c) 23 ns (d) 35 ns

47. The amount of ROM needed to implement a 4 bit multiplier is **[2012, 1 mark]**

(a) 64 bit (b) 128 bit
(c) 1 k bit (d) 2 k bit

48. Register renaming is done in pipelined processors **[2012, 1 mark]**
(a) as an alternative to register allocation at comptile time
(b) for efficient access to function parameters and local variables
(c) to handle certain kinds of hazards
(d) as part of address translation

49. In a k-way set associative cache, the cache is divided into v sets, each of which consists of k lines. The lines of a set are placed in sequence one after another. The lines in set s are sequenced before the lines in set (s + 1). The main memory blocks are numbered 0 onwards. The main memory block numbered j must be mapped to any one of the cache lines from **[2013, 1 Mark]**
(a) (j mod v) * k to (j mod v) * k + (k – 1)
(b) (j mod v) to (j mod v) + (k – 1)
(c) (j mod k) to (j mod k) + (v – 1)
(d) (j mod k) * v to (j mod k) * v + (v – 1)

50. Consider the following sequence of micro-operations.

MBR ← PC
MAR ← X
PC ← Y
Memory ← MBR

Which one of the following is a possible operation performed by this sequence? **[2013, 2 Marks]**
(a) Instruction fetch
(b) Operand fetch
(c) Conditional branch
(d) Initiation of interrupt service

51. Consider a hard disk with 16 recording surfaces (0-15) having 16384 cylinders (0-16383) and each cylinder contains 64 sectors (0-63). Data storage capacity in each sector is 512 bytes. Data are organised cylinder-wise and the addressing format is < cylinder number, surface number, sector number >. A file of size 42797 KB is stored in the disk and the starting disk location of the file is < 1200, 9, 40 >. What is the cylinder number of the last sector of the file, if it is stored in a contiguous manner? **[2013, 2 Marks]**
(a) 1281 (b) 1282
(c) 1283 (d) 1284

52. A RAM chip has a capacity of 1024 words of 8 bits each (1K × 8). The number of 2 × 4 decoders with enable line needed to construct a 16K × 16 RAM from 1K × 8 RAM is **[2013, 2 Marks]**
(a) 4 (b) 5
(c) 6 (d) 7

53. Consider an instruction pipeline with five stages without any branch prediction: Fetch Instruction (FI), Decode Instruction (DI), Fetch Operand (FO), Execute Instruction (EI) and Write Operand (WO). The stage delays for FI, DI, FO, EI and WO are 5 ns, 7 ns, 10 ns, 8 ns and 6 ns, respectively. There are intermediate storage buffers after each stage and the delay of each buffer is 1 ns. A program consisting of 12 instructions I_1, I_2, I_3, ..., I_{12} is executed in this pipelined processor. Instruction I_4 is the only branch instruction and its branch target is I_9. If the branch is taken during the execution of this program, the time (in ns) needed to complete the program is **[2013, 2 Marks]**
(a) 132 (b) 165
(c) 176 (d) 328

Common Data for Questions 54 and 55:
The following code segment is executed on a processor which allows only register operands in its instructions. Each instruction can have atmost two source operands and one destination operand. Assume that all variables are dead after this code segment. **[2013, 2 Marks]**

```
c = a + b;
d = c * a;
e = c + a;
x = c * c;
if (x > a) {
    y = a * a;
}
else {
    d = d * d;
    e = e * e;
}
```

54. Suppose the instruction set architecture of the processor has only two registers. The only allowed compiler optimization is code motion, which moves statements from one place to another while preserving correctness. What is the minimum number of spills to memory in the compiled code? **[2013, 2 Marks]**
(a) 0 (b) 1
(c) 2 (d) 3

55. What is the minimum number of registers needed in the instruction set architecture of the processor to compile this code segment without any spill to memory? Do not apply any optimization other than optimizing register allocation. **[2013, 2 Marks]**
(a) 3 (b) 4
(c) 5 (d) 6

Statement for Linked Answer Questions 56 and 57:
A computer uses 46-bit virtual address, 32-bit physical address and a three-level paged page table organisation. The page table base register stores the base address of the first-level table (T_1), which occupies exactly one page. Each entry of T_1 stores the base address of a page of the second-level table (T_2). Each entry of T_2 stores the base address of a page of the third-level table (T_3). Each entry of T_3 stores a page table entry (PTE). The PTE is 32 bits in size. The processor used in the computer has a 1 MB 16-way set associative virtually indexed physically tagged cache. The cache block size is 64 bytes.

56. What is the size of a page in KB in this computer? **[2013, 2 Marks]**
(a) 2 (b) 4
(c) 8 (d) 16

57. What is the minimum number of page colours needed to guarantee that no two synonyms map to different sets in the processor cache of this computer? **[2013, 2 Marks]**

(a) 2 (b) 4
(c) 8 (d) 16

58. A machine has a 32-bit architecture, with 1-word long instructions. It has 64 registers, each of which is 32 bits long. It needs to support 45 instructions, which have an immediate operand in addition to two register operands. Assuming that the immediate operand is an unsigned integer, the maximum value of the immediate operand is _____ . **[2014, Set-1, 1 Mark]**

59. Consider a 6-stage instruction pipeline, where all stages are perfectly balanced. Assume that there is no cycle-time overhead of pipelining. When an application is executing on this 6-stage pipeline, the speedup achieved with respect to non-pipelined execution if 25% of the instructions incur 2 pipeline stall cycles is _____. **[2014, Set-1, 2 Marks]**

60. An access sequence of cache block addresses is of length N and contains n unique block addresses. The number of unique block addresses between two consecutive accesses to the same block address is bounded above by k. What is the miss ratio if the access sequence is passed through a cache of associativity $A \geq k$ exercising least-recently-used replacement policy? **[2014, Set-1, 2 Marks]**

(a) n/N (b) 1/N
(c) 1/A (d) k/n

61. 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 number of bits for the TAG field is _____. **[2014, Set-2, 1 Mark]**

62. In designing a computer's cache system, the cache block (or cache line) size is an important parameter. Which one of the following statements is correct in this context? **[2014, Set-2, 2 Marks]**

(a) A smaller block size implies better spatial locality
(b) A smaller block size implies a smaller cache tag and hence lower cache tag overhead
(c) A smaller block size implies a larger cache tag and hence lower cache hit time
(d) A smaller block size incurs a lower cache miss penalty

63. If the associativity of a processor cache is doubled while keeping the capacity and block size unchanged, which one of the following is guaranteed to be NOT affected? **[2014, Set-2, 2 Marks]**

(a) Width of tag comparator
(b) Width of set index decoder
(c) Width of way selection multiplexor
(d) Width of processor to main memory data bus

64. The value of a *float* type variable is represented using the single-precision 32-bit floating point format of IEEE-754 standard that uses 1 bit for sign, 8 bits for biased exponent and 23 bits for mantissa. A *float* type variable X is assigned the decimal value of –14.25. The representation of X in hexadecimal notation is **[2014, Set-2, 2 Marks]**

(a) C1640000H (b) 416C0000H
(c) 41640000H (d) C16C0000H

65. Consider a main memory system that consists of 8 memory modules attached to the system bus, which is one word wide. When a write request is made, the bus is occupied for 100 nanoseconds (ns) by the data, address, and control signals. During the same 100 ns, and for 500 ns thereafter, the addressed memory module executes one cycle accepting and storing the data. The (internal) operation of different memory modules may overlap in time, but only one request can be on the bus at any time. The maximum number of stores (of one word each) that can be initiated in 1 millisecond is _____ **[2014, Set-2, 2 Marks]**

66. Consider the following processors (ns stands for nanoseconds). Assume that the pipeline registers have zero latency.

P1: Four-stage pipeline with stage latencies 1 ns, 2 ns, 2 ns, 1 ns.
P2: Four-stage pipeline with stage latencies 1 ns, 1.5 ns, 1.5 ns, 1.5 ns.
P3: Five-stage pipeline with stage latencies 0.5 ns, 1 ns, 1 ns, 0.6 ns, 1 ns.
P4: Five-stage pipeline with stage latencies 0.5 ns, 0.5 ns, 1 ns, 1 ns, 1.1 ns.

Which processor has the highest peak clock frequency? **[2014, Set-3, 1 Mark]**

(a) P1 (b) P2
(c) P3 (d) P4

67. An instruction pipeline has five stages, namely, instruction fetch (IF), instruction decode and register fetch (ID/RF), instruction execution (EX), memory access (MEM), and register writeback (WB) with stage latencies 1 ns, 2.2 ns, 2 ns, 1 ns, and 0.75 ns, respectively (ns stands for nanoseconds). To gain in terms of frequency, the designers have decided to split the ID/RF stage into three stages (ID, RF1, RF2) each of latency 2.2/3 ns. Also, the EX stage is split into two stages (EX1, EX2) each of latency 1 ns. The new design has a total of eight pipeline stages. A program has 20% branch instructions which execute in the EX stage and produce the next instruction pointer at the end of the EX stage in the old design and at the end of the EX2 stage in the new design. The IF stage stalls after fetching a branch instruction until the next instruction pointer is computed. All instructions other than the branch instruction have an average CPI of one in both the designs. The execution times of this program on the old and the new design are P and Q nanoseconds, respectively. The value of P/Q is _____. **[2014, Set-3, 2 Marks]**

68. The memory access time is 1 nanosecond for a read operation with a hit in cache, 5 nanoseconds for a read operation with a miss in cache, 2 nanoseconds for a write operation with a hit in cache and 10 nanoseconds for a write operation with a miss in cache. Execution of a sequence of instructions involves 100 instruction fetch operations, 60 memory operand read operations and 40 memory operand write operations. The cache hit-ratio is 0.9. The average memory access time (in nanoseconds) in executing the sequence of instructions is _____. **[2014, Set-3, 2 Marks]**

Practice Exercise

1. The most appropriate matching for the following pairs :

X :	Indirect addressing	1.	Loop
Y :	Intermediate addressing	2.	Pointers
Z :	Auto decrement addressing	3.	constants

(a) X-3, Y-2, Z-1 (b) X-1, Y-3, Z-2
(c) X-2, Y-3, Z-1 (d) X-3, Y-1, Z-2

2. Addressing mode facilitate access to an operand whose location is defined in relative to the beginning of the data structure in which it appears
(a) absolute (b) immediate
(c) index (d) indirect

3. Which of the following addressing modes are suitable for program relocation time?
1. Absolute addressing 2. Based addressing
3. Relative addressing 4. Indirect addressing
(a) 1 and 4 (b) 1 and 2
(c) 1, 2 and 4 (d) 2 and 3

4. The following program starts at location 0100 H

```
LX1 SP, 00FF
LXI  H, 0701
A,  20H
```

The content of accumulator when the program counter reaches 0109 H is
(a) 20 H (b) 02 H
(c) 00 H (d) FF H

5. Which of the following instructions is an example of direct addressing mode?
(a) MOV A, B (b) 2050
(c) 05 (d) HLT

6. If a cache access requires one clock cycle and handling cache misses stalls the processor for an additional five cycles, which of the following cache hit rates comes closest to achieving an average memory access of 2 cycles?
(a) 75% (b) 80%
(c) 83% (d) 86%

7. An instruction is stored at location 300 with its address field at location 301 the address field has value 400. A processor register R1 contains the number 200. Evaluate the effective address, matching the following addressing modes to their respective address.

a.	Direct	1.	702
b.	Immediate	2.	200
c.	Relative	3.	400
d.	Register indirect	4.	600
e.	Index (R1 is indirect)	5.	300

(a) a-3, b-5, c-1, d-2, e-4 (b) a-3, b-4, c-2, d-1, e-5
(c) a-3, b-3, c-2, d-1, e-4 (d) a-4, b-3, c-1, d-5, e-2

8. Match the following in context of an 8085 µP

A.	DAA	1.	Program control instruction
B.	LXI	2.	Data movement instruction
C.	RST	3.	Interrupt instruction
D.	JMP	4.	Instruction

Codes

	A	B	C	D
(a)	1	2	3	4
(b)	4	2	3	1
(c)	3	2	1	4
(d)	2	3	4	1

9. Stack adressing is same as
(a) relative addressing (b) zero addressing
(c) virtual addressing (d) indirect addressing

10. Match the following :

a.	Immediate address mode	1.	local variables
b.	Direct address mode	2.	Relocatable program
c.	Indirect, address mode	3.	Pointers
d.	Index address mode	4.	Locality of references
e.	Base address mode	5.	Arrays
f.	Relative address mode	6.	Constant operands

(a) a-6, b-1, c-3, d-5, e-2, f-4
(b) a-5, b-4, c-6, d-3, e-1, f-2
(c) a-3, b-5, c-2, d-4, e-1, f-6
(d) a-6, b-5, c-2, d-3, e-1, f-4

11. Programming that actually controls the path of the data within the computer is called
(a) micro programming
(b) system programming
(c) assemble language
(d) machine language programming

12. The control signal indicates
(a) whether a data is read into or written out to memory
(b) whether CPU is accessing memory of input/output device
(c) whether input/output device or memory is ready to transfer data
(d) All of the above

13. LRU is an effective cache replacement strategy primarily because programs
(a) exhibit locality of reference
(b) usually have small working sets
(c) read data much more frequently than write data
(d) None of these

14. The CPU of a computer takes instruction from the memory and executes them. This process is called
(a) load cycle (b) time sequencing
(c) fetch-execute cycle (d) clock cycle

15. An interrupt can be temporarily ignored by the counter is called
(a) vectored interrupt (b) non-maskable interrupt
(c) maskable interrupt (d) low priority interrupt

16. Match List I with List II and select the correct answer from the codes given below the lists.

	List I		**List II**
A	A shift register can be used	1.	for code conversion
B.	A multiplexer can be used	2.	to generate memory chip select
C.	A decoder can be used	3.	for parallel to serial conversion
		4.	as many to one switch
		5.	for analog to digital conversion

Codes

	A	B	C		A	B	C
(a)	3	1	2	(b)	4	3	2
(c)	3	4	2	(d)	2	3	4

17. When a subroutine is called, then address of the instruction following the CA1 instruction is stored in the
(a) stack pointer (b) program counter
(c) accumulator (d) stack

18. The sequence of events that happens during a typical fetch operation is
(a) PC → MAR → Memory → MDR → 1R
(b) PC → Memory → MDR → 1R
(c) PC → Memory → 1R
(d) PC → MAR → Memory → 1R

19. Any instruction should have atleast
(a) 2 operands (b) 1 operands
(c) 3 operands (d) None of these

20. In ..... the total address space exceeds the size of physical memory.
1. paging
2. segmentation
(a) 1, 2 (b) 1
(c) 2 (d) None of these

21. The ability temporarily halt the CPU and use this time to send information on buses is called
(a) direct memory access (b) vectoring the interrupt
(c) palling (d) cycle stealing

22. Microprogram is
(a) the name of source program in micro computers
(b) the set of instructions indicating the primitive operations in a system
(c) primitive form of macros used in assembly language programming
(d) program of very small size

23. Which of the following units is responsible for coordinating various operations using timing signals?
(a) ALU (b) control unit
(c) Memory unit (d) I/O unit

24. The register which contains the data to be written into or read out of the addressed location is called
(a) memory addressed register
(b) memory data register
(c) program counter
(d) index register

25. Microprogramming is a technique for
(a) writing small programs effectively
(b) programming output/input routines
(c) programming the microprocessor
(d) programming the control steps of a computer

26. The device which is used to connect a peripheral to bus is called
(a) control register (b) communication protocol
(c) interface (d) None of the above

27. Comparing the time T_1 taken for a single instruction on a pipelilned CPU with time T_2 taken on a non-pipelined but identical CPU, we can say that
(a) $T_1 \leq T_2$
(b) $T_1 \geq T_2$
(c) $T_1 < T_2$
(d) T_1 is T_2 plus line taken for one instruction fetch cycle

28. Branch instructions are handled in pipeline usinig
(a) prefetch target instruction strategy
(b) loop Greffer strategy
(c) Both (a) and (b)
(d) None of the above

29. What is the speed up factor of n-stage pipelined
(a) (n + 1) (b) (n – 1)
(c) 2n (d) n

30. A computer system has a 4k word cache organized in blocked set associative manner with 4 blocks per set, 64 words per block. The numbers of bits in the SET and WORD fields of the main memory address format are
(a) 15, 40 (b) 7, 2
(c) 6, 4 (d) 4, 6

31. What is the minimum size of ROM required to store the complete truth table of an 8-bit × 8-bit multiplier?
(a) 32k × 16 bit (b) 64k × 16 bit
(c) 16k × 32 bit (d) 64k × 32 bit

32. A hard disk has 63 sectors per stract, 10 platters each with 2 recording surface and 1000 cylinders. The address of a sector is given as a triple <c, h, c>, where c is the cylinder number, h is the surface number and s is the sector number. Thus, the 0th sector is addressed as <0, 0, 0>, the 1st sector as < 0, 0, 1> and so on
(a) 50 50 35 (b) 50 50 36
(c) 50 50 37 (d) 50 50 38

33. Consider the following register-transfer language

$$R_3 \longleftarrow R_3 + M(R_1 + R_2)$$

where R_1, R_2 are the CPU registers and M is a memory location in primary memory. Which addressing mode is suitable for above register-transfer language?
(a) immediate (b) indexed
(c) direct (d) displacement

34. In a 16-bit instruction code format 3 bit operation code, 12 bit address and 1 bit is assigned for address mode designation. For indirect addressing, the mode bit is
(a) 0 (b) 1
(c) pointer (d) off-set

35. What does CISC and RISC means?
(a) common instruction set controller and rare instruction set controller
(b) complex instruction set controller and reduced instruction set controller
(c) compiled instruction set source code and recompiled instruction source code
(d) None of the above

36. The computer performs all mathematical and logical operations inside its
(a) memory unit (b) central processing unit
(c) output unit (d) visual display unit

37. Match List-I with List-II and select the correct answer using the codes given below the lists :

	List-1	**List-II**
A.	$Regs[R_4] \leftarrow Regs[R_4] + Regs[R_3]$	1. immediate
B.	$Regs[R_4] \leftarrow Regs[R_4] + 3$	2. register
C.	$Regs[R_4] \leftarrow Regs[R_4] + Mem[Regs[R_1]]$	3. displacement

Codes

	A	B	C
(a)	3	2	1
(b)	2	1	3
(c)	1	2	3
(d)	3	1	2

38. Pipelining improves CPU performance due to
(a) reduced memory access time
(b) increased clock speed
(c) the introduction of parallelism
(d) additional funcional units

39. A 32-bit address bus allows access to a memory of capacity
(a) 64 Mb (b) 6 Mb
(c) 1 Gb (d) 4 Gb

40. The read/write line
(a) belongs to the data bus
(b) belongs to the control bus
(c) belongs to the address bus
(d) CPU bus

41. Which of the following lists memory types from highest lowest access speed?
(a) secondary storage, main memory, cache, registers
(b) registers, cache, secondary storage, main memory
(c) registers, cache, main memory, secondary storage
(d) cache, registers, main memory, secondary storage

42. _____ improve system performance by temporarily storing data during transfers between devices processes that operate at different speeds.
(a) cache (b) controllers
(c) buffers (d) registers

43. Which of following registers processor used for fetch and execute operations?
1. Program counter 2. Instruction register
3. Address register
(a) 1 and 3 (b) 1 and 2
(c) 2 and 3 (d) 1, 2 and 3

44. Consider an (n + k) bit instruction with a k-bit opcode and single n-bit address. Then this instruction allow _____ operations and _____ addressable memory cells.
(a) 2^k, 2^{n+k} (b) 2^{n+k}, 2^{n+k}
(c) 2^k, 2^n (d) 2^{n+k}, 2^{n+1}

45. Which of the following statements is false with regard to fully associative and direct mapped cache organizations?
(a) Direct mapped caches do not need a cache block replacement policy, whereas fully associative caches do.
(b) In a fully associative cache, the full address is stored alongside the data block.
(c) A direct mapped cache is organized according to an 'index', and the 'tag' part of the memory address is stored alongside the data block.
(d) Direct mapped caches may produce more misses if programs refer to memory words that occupy a single tag value.

46. The register which holds the address of the location to or from which data are to be transferred is known as
(a) index register
(b) instruction register
(c) memory address register
(d) memory data register

47. If increasing the block size of a cache improves performance it is primarily because programs
(a) exhibit locality of reference
(b) usually have small working sets
(c) read data much more frequently than write data
(d) None of these

48. Consider the following program:

```
integer A[1000];
for i = 1 to 1000
  for j = 1 to 1000
    A[i] = A[i] + 1
```

When the above program is compiled with all compiler optimizations turned off and run on a processor with a 1K byte fully-associative write-back data cache with 4-word cache blocks, what is the approximate data cache miss rate? (Assume integers are one word long and a word is 4 bytes.)
(a) 0.0125% (b) 0.05%
(c) 0.1% (d) 5%

49. Match List-I with List-II using memory hierarchy from top to down and select the correct answer using the codes given below the lists.

	List-I	**List-II**
A.	Size	1. Increases
B.	Cost/bit	2. Decreases
C.	Access time	3. No change
D.	Frequency of access	

Codes

	A	B	C	D
(a)	1	1	2	2
(b)	2	1	2	1
(c)	2	2	1	1
(d)	1	2	1	2

50. A computer system has 4K-word cache organized in a block-set-associative manner, with 4 blocks per set, 64 words per block. Memory is word addressable. The number of bits in the SET and WORD fields of the main memory address format is

(a) 15, 4 (b) 6, 4
(c) 7, 6 (d) 4, 6

51. Which of the following units can be used to measure the speed of a computer

(a) SYPS (b) MIPS
(c) BAUD (d) None of these

52. Consider the following I/O instruction format for IBM 370 I/O channel

Operation Code	Channel Address	Device Address

Then operation code specifies

1. test I/O
2. test channel
3. store channel identification
4. halt device

(a) Only 1 and 4 (b) Only 2 and 3
(c) 1, 2, 3 and 4 (d) 1, 3 and 4

53. Match List-I with List-II and select the correct answer using the codes given below the lists :

	List-I	**List-II**
A.	0-address instruction	1. T = TOP (T – 1)
B.	1-address instruction	2. Y = Y + X
C.	2-address instruction	3. Y = A – B
D.	3-address instruction	4. ACC = ACC – X

Codes

	A	B	C	D
(a)	1	2	3	4
(b)	3	2	4	1
(c)	2	3	1	4
(d)	1	4	2	3

54. Find the average access time experienced by the CPU in a system with two levels of caches.

If h_1 : is the hit rate in the primary cache
h_2 : is the hit rate in the secondary cache
c_1 : is the time to access information in the primary cache
c_2 : is the time to access information in the secondary cache
M : is the time to access information in the main memory

(a) $h_1c_1 + h_2c_2 + (1 - h_1h_2)M$
(b) $h_1c_1 - h_2c_2 + (1 - h_1h_2)M$
(c) $h_1c_1 + (1 - h_2)h_1c_2 + (1 - h_1)(1 - h_2)M$
(d) $h_1c_1 + (1 - h_1)h_2c_2 + (1 - h_1)(1 - h_2)M$

55. The following are three statements about the "locality of reference" principle used in memory system –

1. It says that it is more likely that an already accessed memory location is accessed further, and it is more likely that surrounding (adjacent) memory locations will also be accessed.
2. It is an observation that equally widely used to implement virtual memory and cache memory systems.
3. It syas that the bytes of a word be interleaved on several physical memory modules for better performance.

Which of them is/are true?

(a) Only 1 is true (b) Only 2 is true
(c) Only 3 is true (d) Only 1 nad 2 are true

56. The sequence of events that happen during a typical fetch operation is

(a) PC → MAR → Memory → MDR → IR
(b) PC → Memory → MDR → IR
(c) PC → Memory → IR
(d) PC → MAR → Memory → IR

57. A cache has a hit rate of 95 percent, 128 byte lines and a cache hit latency of 5ns. The main memory takes 100 ns to return the first word (32 bits) of a line, and 10 ns to return each subsequent word.

What is T_{mins} for this cache ? (Assume that the cache waits until the line has been fetched into the cache and then reexecutes the memory operation, resulting in a cache hit. Neglect the time required to write the lines into the cache once has been fetched from the main memory. Also assume that the cache takes the same amount of time to detect that a mass has occured as to handle a cache hit.)

(a) 410 ns (b) 420 ns
(c) 240 ns (d) 540 ns

58. Both's coding in 8-bits for the decimal number –57 is _____.

(a) 0 – 100 + 1000 (b) 0 – 100 + 100 – 1
(c) 0 – 1 + 100 – 10 + 1 (d) 00 – 10 + 100 – 1

59. Daisy Chained and independent Request bus arbitration requires _____ respectively. (Where N stands for number of Bus Master Devices in the configuration).

(a) 2 and 7 signal lines
(b) 3 and 2^N signal lines
(c) 2N and 2N signal lines
(d) 3 and 2N signal lines

60. In which addressing mode, the effective address of the operand in gererated by adding a constant value to the contents of a register?

(a) absolute mode (b) indirect mode
(c) immediate mode (d) index mode

61. Match List-I with List-II and select the correct answer using the codes given below the lists.

	List I	**List II**
A.	A[1] = B[1];	1. indirect addressing
B.	while [*A++];	2. indexed addressing
C.	int temp = *X	3. Autoincrement

Codes

	A	B	C
(a)	3	2	1
(b)	1	3	2
(c)	2	3	1
(d)	1	2	3

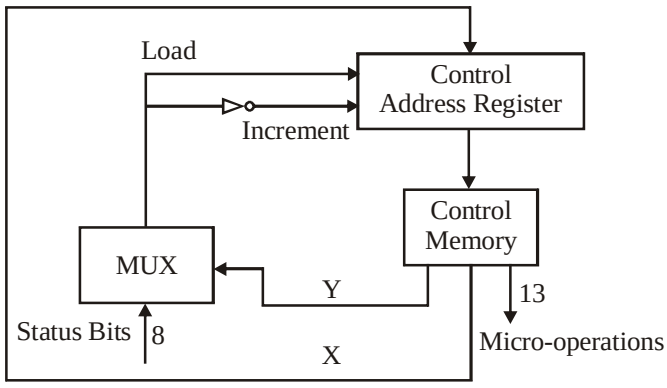
62. The memory locations 1000, 1001 and 1020 have data values 18, 1 and 16 respectively before the following program is executed

MOVI Rs, 1;	Move immediate
LOAD Rd. 1000(Rs);	Load from memory
ADD Rd, 1000;	Add immediate
STORE (Rd), 20;	Store immediate

Which of the statements below is TRUE after the program is executed ?

(a) memory location 1000 has value 20
(b) memory location 1020 has value 20
(c) memory location 1021 has value 20
(d) memory location 1001 has value 20

63. The microinstructions stored in the control memory of a processor have a width of 26 bits. Each microinstruction is divided into three fields; a mono-operation field of 13 bits, a next address field (X) and a MUX select field (Y). There are 8 status bits in the inputs of the MUX.



How many bits are there in the X and Y fields, and what is the size of the control memory in number of words?

(a) 10, 3, 1024 (b) 8, 5, 256
(c) 5, 8, 2048 (d) 10, 3, 512

64. Microinstruction length is determined by _____

1. The maximum number of simultaneous micro operations that must be specified.
2. The way in which the control information is represented or encoded.
3. The way in which the next microinstruction address is specified.

(a) 1 and 2 (b) 2 and 3
(c) 1 and 3 (d) All of the above

65. Suppose a cache is 10 times faster than main memory and suppose that the cache can be used 70% of the time. How much speed up do we gain by using the cache?

(a) 0.37 (b) 2.7
(c) 3.7 (d) 0.27

66. Following are some statements associated with microprocessors. Identify the false statement.

(a) The control unit is the component of the CPU that implements the microprocessors instruction set.
(b) The instruction received by the CPU is decoded by the arithmetic logic unit.
(c) A CPU register is used to hold a binary value temporarily for storage, for manipulation, and/or for simple calculations.
(d) The control units are programmed and not hardwired.

67. The following are four statements about Reduced Instruction Set Computer (RISC) architectures.

1. The typical RISC machine instruction set is small, and is usually a subject of a CISC instruction set.
2. No arithmetic or logical instruction can refer to the memory directly.
3. A comparatively large number of user registers are available.
4. Instructions can be easily decoded through hard-wired control units.

68. The performance of a pipelined processor suffers if _____.

(a) the pipeline stages have different delays
(b) consecutive instructions are dependent on each other
(c) the pipeline stages share hardware resources
(d) all of the above

69. Consider the following organization of main memory and cache memory.

Main memory: 64K × 16

Cache memory: 256 × 16

Memory is word-addressable and block size is of 8 words. Determine the size of tag field if direct mapping is used for transforming data from main memory to cache memory.

(a) 5 bits (b) 6 bits
(c) 7 bits (d) 8 bits

70. Match List-I with List-II and select the correct answer using the codes given below the lists:

List - I	**List - II**
A. Stack overflow	1. Software interrupt
B. Supervisor call	2. Internet interrupt
C. Invalid opcode	3. External interrupt
D. Timer	4. Machine check interrupt

	A	B	C	D
(a)	2	3	4	1
(b)	2	1	4	3
(c)	3	1	2	4
(d)	3	1	4	2

71. Consider a CPU has 8 general-purpose registers R_0, R_1, ..., R_7 and supports the following operation.

ADD R_a, R_b, R_c Add R_a to R_b and store the result to R_c.

MUL R_a, R_b, R_c Multiply R_a to R_b and store the result to R_c.

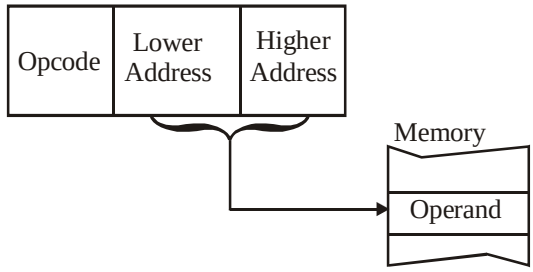
An operation normally takes one clock cycles, an operation takes two clock cycles if it produces a result required by the immediately following operations. Consider the expression XY + XYZ + YZ, where variables X, Y and Z are initially located in the registers R_0, R_1 and R_2. If contents these registers must not be modified, what is the minimum number of clock cycles required for an operation sequence that computes the value of XY + XYZ + YZ?

(a) 4 (b) 5
(c) 6 (d) 7

72. In a two level memory hierarchy, the access time of the cache memory is 12 nanoseconds and the access time of the main memory is 1.5 microseconds. The hit ratio is 0.98. What is the average access time of the two level memory system?
(a) 13.5 nsec (b) 42 nsec
(c) 7.56 nsec (d) 41.76 nsec

73. The word length of a CPU is defined as
(a) the maximum addressable memory size
(b) the width of a CPU register (integer or float point)
(c) the width of the adress bus
(d) the number of general purpose CPU registers

74. The following diagram shows, which addressing mode?



(a) immediate addressing mode
(b) indirect addressing mode
(c) extended addressing mode
(d) none of the above

75. A certain computer system design has a single CPU, a two-level cache, and supports memory mapped I/O for output-only controllers. Which of the following is true?
(a) The design i impossible, since memory mapped I/O will prevent cache coherence.
(b) In two-level caches, the L_1 cache is generally built from SRAM.
(c) In two-level caches, the L_1 cache is generally larger than L_2 cache.
(d) In two-level caches, the L_2 cache generally has a lower latency than the L_1 cahche.

76. Consider the following sequence of instructions intended for execution on a stack machine. Each arithmetic operation pops the second operand, then pops the first operand, operates on them, and then pushes the result back onto the stack

```
push   b
push   x
add
pop    c
push   c
push   y
add
push   c
sub
pop    z
```

Which of the following statements is/are true?
1. If push and pop instructions each require 5 bytes of storage, and arithmetic operations each require 1 byte of storage then the instruction sequence as a whole requires a total of 40 bytes of storage.
2. At the end of execution, z contains the same value as y.
3. At the end of execution, the stack is empty.

(a) 1 only (b) 2 only
(c) 2 and 3 only (d) 1 and 3 only

77. A CPU has an arithmetic unit that adds bytes and then sets its V, C and Z flag bits as follows. The V-bit is set if arithmetic overflow occurs (in 2's complement arithmetic). The C-bit is set if a carryout is generated from the most significant bit during an operation. The Z-bit is set if the result is zero. What are the values of the V, C and Z flag bit after 8-bit byte 1100 1100 and 1000 1111 are added?

	V	C	Z
(a)	0	0	0
(b)	1	1	0
(c)	1	1	1
(d)	0	1	0

78. The following are the some of the sequences of operations in instruction cycle, which one is correct sequence?
(a) : PC $\rightarrow$ Address register
Data from memory $\rightarrow$ Data register
Data register $\rightarrow$ IR
PC + 1 $\rightarrow$ PC
(b) : Address register $\rightarrow$ PC
Data register $\rightarrow$ Data from memory
Data register $\rightarrow$ IR
PC + 1 $\rightarrow$ PC
(c) : Data from memory $\rightarrow$ Data register
PC $\rightarrow$ Address register
Data register $\rightarrow$ IR
PC + 1 $\rightarrow$ PC
(d) None of these

79. Match List-I with List-II and select the correct answer using the codes given below the lists:

List-I	**List-II**
A. MOV X, R1	1. Three-address instruction
B. STORE X	2. Zero-address instruction
C. POPX	3. Three-address instruction
	4. Two-address instruction

	A	B	C
(a)	4	3	2
(b)	3	2	1
(c)	2	3	4
(d)	None of these		

80. Pipelined operation is interrupted whenever two operations being attempted in parallel need same hardware component. Such a conflict can occur if _____.
(a) The execution phase of an instruction requires access to the main memory, which also contains the next instruction that should be fetched at the same time
(b) The prefetch phase of an instruction requires access to the main memory, which also contains the next instruction that should be fetched at the same time.

(c) The decode phase of an instruction requires access to the main memory, which also contains the next instruction that should be fetched at the same time.
(d) None of these

81. Consider the unpipelined machine with 10 ns clock cycles. It uses four cycles for ALU operations and branches, whereas five cycles for memory operations. Assume that the relative frequencies of there operations are 40%, 20%, 40% respectively. Suppose that due to clock skew and setup, pipelining the machine adds 1 ns overhead to the clock. How much speed up in the instruction executed rate will we gain from a pipeline?
(a) 5 times (b) 8 times
(c) 4 times (d) 4.5 times

82. Consider the following situation and fill in the blanks:
The computer starts the tape moving by issuing a comman; the processor then monitors the status of the tape by means of a _____. When the tape is in the correct position, the processor issues a _____.
(a) Data input, status, data output command
(b) Data input, status control command
(c) Control, status, data output command
(d) Control, status, data input command

83. Asynchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted, way of achieving this _____.
(a) Is by means of a strobe pulse from one of the unit
(b) Is by means of handshaking
(c) Both (a) and (b)
(d) None of these

84. For interval arithmetic best rounding technique use is _____ .
(a) Rounding to plus and minus infinity
(b) Rounding to zero
(c) Rounding to nearest
(d) None of these

85. The percentage of times that a page number is found in the associative register is called the hit ratio.
Assume it takes 50 nanoseconds to search the associative registers and 100 nanoseconds to access main memory. What is the percentage of slowdown in memory-access time if the hit ratio is 90%?
(a) 24 (b) 40
(c) 45 (d) 60

86. An IOSA defines 2 different instruction formats. In the first format, 6 bits are used for the opcode, among these opcode bit patterns, three are used as special patterns. When any of these three patterns appear in the opcode field, the instruction is encoded using the second format, and the actual opcode is present in another 6-bit field. How many unique opcodes can this ISA support?
(a) 64 (b) 253
(c) 254 (d) 189

87. Consider the following three statements.
1. A multilevel view of a computer system simplifies the design of a complex system by identifying specific functionalities for each level.
2. The Instruction Set Architecture (ISA) level defines the characteristics of the CPU.
3. One of the functions of the operating system machine level is to enfore a separation between the users tasks and the privileged tasks.

Which of the following statement(s) is/are true about the multilevel machine view point?
(a) Only 1 is true (b) Only 1 and 3 are true
(c) Only 3 is true (d) All are true

88. In a general purpose computer system the CPU, the main memory and the cache may be interconnected via one or more shared system bus(es). However, input/output devices (eg. Hard disk, network interfaces) may only be connected to the system bus through an I/O controller. The following are four statements regarding the requirement for an I/O controller.
1. The capacities of I/O devices are magnitude order larger than that of main memory and hence direct interfacing is impossible.
2. The response times of I/O devices are magnitude order slower than that of CPU and hence direct interfacing is impossible.
3. It is always better to off load the I/O processing to a secondary processor on the I/O controller board then to depend on the primary CPU for I/O processing.
4. The variety of I/O devices in the market requires that a separate I/O controller exist for each device.

What statement(s) best explain the requirement for an I/O controller?
(a) Only 1 is true (b) Only 3 is true
(c) Only 2 and 3 are true (d) Only 2, 3 and 4 are true

89. A certain processor executes the following set of machine instructions sequentially.
MOV R_0 # 0
MOV R_1, 100 (R_0)
ADD R_1, 200 (R_0)
MOV 100 (R_0), R_1
Assuming that memory location 100 contains the value 35 (Hex), and the memory location 200 contains the value A4 (Hex), what could be said about the final result?
(a) Memory location 100 contains value A4
(b) Memory location 100 contains value D4
(c) Memory location 100 contains value D9
(d) Memory location 200 contains value 35

90. An 8-Bit DMA Device is operating is Cycle Stealing Mode (Single Transfer Mode). Each DMA cycle is of 6 clock states and DMA clock is 2 MHz. Intermediate CPU machine cycle takes 2 μS, determine the DMA Data Transfer Rate
(a) 100 Kbytes/sec (b) 200 Kbytes/sec
(c) 350 Kbytes/sec (d) None of the above

91. Determine the width of Micro-instruction having following Control signal field, in a Vertical Micro-programmed Control Unit
1. Next address field of 7 bits
2. ALU function field selecting 1 out of 13 ALU function
3. Register-in field selecting 1 out of 13 ALU function

4. Register-out field selecting 1 out of 8 registers
5. Shifter field selecting no shift, right shift or left shift
6. Auxiliary control field of 4 bits

(a) 22 (b) 23
(c) 24 (d) 25

92. Which one of the following is true for a CPU having a single interrupt request line and a single interrupt grant line?
(a) Neither vectored interrupt nor multiple interrupting devices are possible
(b) Vectored interrupts are not possible but multiple interrupting devices are possible
(c) Vectored interrupts and multiple interrupting devices are both possible.
(d) Vectored interrupts is possible but multiple interrupting devices are not possible.

93. When multiplicand Y is multiplied by multiplier $X = x_{n-1} x_{n-2} \cdots x_0$ using bit-pair recording in Booth's algorithm, partial products are generated according to the following table.

Row	x_{i+1}	x_i	x_{i-1}	Partial product
1	0	0	0	0
2	0	0	1	Y
3	0	1	0	Y
4	0	1	1	2Y
5	1	0	0	?
6	1	0	1	– Y
7	1	1	0	– Y
8	1	1	1	?

The partial products for rows 5 and 8 are
(a) 2Y and Y (b) –2Y and 2Y
(c) –2Y and 0 (d) 0 and Y

94. A cache contains n blocks and main memory contains m blocks. If k way set associative maping is used than what will be number of TAG bits–

(a) $\log_2 \frac{mk}{n}$ (b) $\log_2 \frac{m}{n}$

(c) $\log_2 \frac{nk}{m}$ (d) $\log_2 \frac{mn}{k}$

95. In a vectored interrupt
(a) the branch address is assigned to a fixed location in memory
(b) the interrupting source supplies the branch information to the processor through an interrupt vector
(c) the branch address is obtained from a register in the processor
(d) none of the above

96. A device with data transfer rate 10 KB/sec is connected to a CPU. Data is transferred byte-wise. Let the interrupt overhead be 4 μsec. The byte transfer time between the device interface register and CPU or memory is negligible. What is the minimum performance gain of operating the device under interrupt mode over operating it under program controlled mode?
(a) 15 (b) 25
(c) 35 (d) 45

97. Consider a disk drive with the following specifications: 16 surfaces, 512 tracks/surface, 512 sectors/track, 1 KB/sector, rotation speed 3000 rpm. The disk operated in cycle stealing mode whereby whenever one byte word is ready it is sent to the memory; similarly, for writing, the disk interface reads a 4 byte word from the memory in each DMAcycle. Memory cycle time is 40 nsec. The maximum percentage of time that the CPU gets blocked during DMA operation is
(a) 10 (b) 25
(c) 40 (d) 50

98. We have two designs D1 and D2 for a synchronous pipeline processor. D1 has 5 pipeline stages with execution times of 3 nsec, 2 nsec, 4 nsec, 2 nsec and 3 nsec while the design D2 has 8 pipeline stages each with 2 nsec execution time. How much time can be saved using design D2 over design D1 for executing 100 instructions?
(a) 214 nsec (b) 202 nsec
(c) 86 nsec (d) 200 nsec

99. Which of the following is true in case of lockup free cache?
1. A cache structure allows a CPU to access a cache while a hit is being serviced.
2. A cache structure allows a CPU to access a cache while a miss is being serviced.
3. A cache that can support multiple outstanding misses.
4. A cache that can support multiple outstanding hits.

(a) Only 1 and 4 (b) Only 2 and 3
(c) Only 1 (d) Only 2

100. Suppose that an unpipelined processor has a cycle time of 25 ns, and that its data path is made up of modules with latencies of 2, 3, 4, 7, 3, 2 and 4 ns (in that order). In pipelining this processor, it is not possible to rearrange the order of the modules (for examples, putting the register read stage before the instruction decode stage) or to divide a module into multiple pipeline stages (for complexity reasons). Given pipeline latches with 1 ns latency: If the processor is divided into the rewest number of stages that allow is to achieve the minimum latency from part 1, what is the latency of the pipeline?
(a) no latency (b) 35 ns latency
(c) 40 ns latency (d) 56 ns latency

NUMERICAL TYPE QUESTIONS

101. A 4-stage pipeline has the stage delays as 150, 120, 160 and 140 ns respectively. Registers that are used between the stages have a delay of 5 ns. Assuming constant locking rate the total time required to process 1000 data items on this pipeline in

102. A non-pipeline sysem takes 40 ns to process a task. The same task can be processed in a 6-segment pipeline with a clock cycle of 10 s. Determine the speed up ratio of the pipeline for 100 tasks.

103. The refreshing rate of dynamic RAMs is in the range of _____µs.

104. Consider a small two-way set associative cache memory, consisting of four blocks, for choosing the block to be replaced. Use the least recently scheme. The number of cache misses for the following sequence of block addresses 8, 12, 0, 12, 8 is

105. In a virtual memory system, size of virtual address is 32-bit, size of physical adress is 30-bit, page size is 4 kb and size of each page table entry in 32-bit. The main memory is byte addressable. Which one of the following is the maximum number of bits that can be used for storing protection and other information in each page table entry?

106. When cache is faster than main memory and cache can be used 90% of the time, how much speed we gain by using the cache?

107. What will be average cost per bit for a system with main memory of 1024 cost, 100 units and secondary memory of 4096 cost, 10 units.

108. An 8 bit register R contain the binary value 10011100 what is the register value after an arithmetic shift right? Starting from the initial number 10011100, determine the register value after an arithmetic shift left. Enter the binary digit.

109. How many memory chips of 128 × 8 are needed to provide memory capacity of 4096 × 16.

110. How many 128 × 8 RAM chips are needed to provide a memory capacity of 2048 bytes?Enter a value.

111. How many lines of the address bus must be used to access 2048 bytes of memory ? Enter a value.

112. If a disk spins at 10,000 rpm what is the average rational latency time of a request? If a given track on the disk has 1024 sectors, what is the transfer time for a sector?

113. In a cache with 64 byte cache lines how may bits are used to determine which byte within a cache line an address points to?

114. For a cache with a capacity of 32 KB, How many lines does the cache hold for line lengths of 32, 64 or 128 bytes?

115. If a cache has a capacity of 16KB and a line length of 128 bytes, how many sets does the cache have if it is 2-way, 4-way, or 8-way set associative?

116. If a cache memory has a hit rate of 75 percent, memory request take l2ns to complete if they hit in the cache and memory request that miss in the cache take 100 ns to complete, what is the average access time of cache?

117. In a two level memory hierarchy, if the cache has an access time of ns and main memory has an access time of 60ns, what is the hit rate in cache required to give an average access time of 10ns?

118. A computer employ chips of 256 × 8 and ROM chips of 1024 × 8. The computer system needs 2k byte of RAM, 4K. bytes of ROM and four interface units, each with four register. A memory mapped 1/0 configuration is used. The two highest order bits of the address assigned 00 for RAM, 01 for ROM, and 10 for interface registers. How many RAM and ROM chips are required?

119. In a non-pipelined single-cycle-per-instruction processor with an instruction cache, the average instruction cache miss rate is 5%. It takes 8 clock cycles to fetch a cache line from the main memory. Disregarding data cache misses, what is the approximate average CPI (cycles per instruction)?

COMMON DATA & LINKED ANSWER TYPE QUESTIONS

Common data for Q. 120 and Q. 121

The mapping function is implemented using the address. For purpose of cache access. each main memory address can be viewed as consisting of three fields.

The least significant 'w' bits identify a unique word or byte within a block of main memory. The remaining 's' bits specify one of the 2^s blocks of main memory. The cache logic interprets these 's' bits as a tag of 's–r' bits and a line field of 'r' bits.

120. With above information, number of lines in cache equals to _____.

(a) 2^r (b) 2^{r+w}

(c) 2^{s+w} (d) undetermined

121. With the above information, the size of tag equals to _____.

(a) r bits (b) (s + w) bits

(c) s bits (d) (s – r) bits

Common data for Qs. 122 to 124.

Consider a machine with a byte addressable main memory of 2^{16} bytes. Assume that a direct mapped data cache consisting of 32 lines of 64 bytes each is used in the system. A 50 × 50 two-dimensional array of bytes is stored in the main memory starting from memory location 1100 H. Assume that the data cache is initially empty. The complete array is accessed twice. Assume that the contents of the data cache do not change in between the two accesses.

122. How many data cache misses will occur in total?

(a) 48 (b) 50

(c) 56 (d) 59

123. Which of the following lines of the data cache will be replaced by new blocks in accessing the array?

(a) line 4 to line 11 (b) line 4 to line 12

(c) line 0 to line 7 (d) line 0 to line 8

124. A cache line is 64 Bytes. The main memory has latency 32 ns and bandwidth 1 G.Bytes/s. The time required to fetch the entire cache line from the main memory is

(a) 32 ns (b) 64 ns
(c) 96 ns (d) 128 ns

Common data for Q. 125 and 126

Let a cache has 126 llines, each with 16 words and is is 2-way set associative. Suppose the main memory has 16-bit address. Both cache and main memory are word addressable.

125. How many bits are in the "tag" field in the cache?

(a) 6 (b) 8
(c) 5 (d) 7

126. Now suppose we want to read or write a word at the memory address 357 A. Under the above set-associative mapping technique what will be the "tag" field in decimal?

(a) 6 (b) 26
(c) 13 (d) 53

Common Data Questions for 127 and 128.

Consider an accumulator-based CPU supports only single address instruction. The CPU supports the following instructions.

LOAD A Load A from memory into accumulator AC.
STORE A Store contents of accumulator AC in memory location M
ADD B Load B into data register (DR) and add to accumulator.
MOV A, B Move content of B to A.
SUB A, B Subtract B from A.

127. Assume CPU uses the memory referencing and each instruction LOAD, STORE and ADD takes on clock cycle. To computer Z = X + Y CPU takes how many minimum number of clock cycles?

(a) 2 (b) 3
(c) 4 (d) 5

128. How many minimum clock cycles are needed when accumulator becomes O. If each instruction takes one clock cycles?

(a) 0 (b) 1
(c) 2 (d) 3

Common Data Questions for 129 and 130.

Suppose a CPU contains 1000 memory references there are 40 misses in L_1 cache (First Level Cache) and 20 misses in the L_2 cache (Second Level Cache). Assume miss penalty from the L_2 cache to memory is 100 clock cycles the hit time of L_2 cache is 10 clock cycles, the hit time of L_1 cache is 1 clock cycle and there are 1.5 memory references per instruction.

129. What is the average memory access time?

(a) 3.4 clock cycles (b) 3.5 clock cycles
(c) 5.3 clock cycles (d) 1.8 clock cycles

130. What is the average memory stalls per instruction?

(a) 3.4 clock cycles (b) 3.5 clock cycles
(c) 3.6 clock cycles (d) 4.5 clock cycles

Common Data Questions for 131 and 132.

Consider an unpipelined processor assume that it has a 1 ns clock cycles and that it uses 4 cycles for ALU operation and branches and 5 cycles for memory operations. Assume that the relative frequencies of these operations are 40%, 20% and 40% respectively. Suppose due to clock skew and setup, pipelining the processor adds 0.2 ns of overhead to the clock. Ignore any latency impact.

131. What is the average instruction execution time for unpipelined processor?

(a) 3.4 ns (b) 4.4 ns
(c) 3.1 ns (d) 1.2 ns

132. What speedup gain after pipelined the processor?

(a) 3.4 ns (b) 3.8 ns
(c) 3.7 ns (d) 1.2 ns

Common Data for Q. 133 and Q 134

Consider the following assembly language program for a hypothetical processor. A, B and C are 8 bit integers. The meaning of various instructions are shown as comments.

MOV B, # 0;	B ← 0
MOV C, # 8;	C ← 8
Z : CMP C, # 0;	compare C with 0
JZX;	jump to X if zero flag is set
SUBC, # 1;	C ← C – 1
LRCA, # 1;	Left rotate A through carry by one bit. Thus if the initial value of A and the carry flag are a7a6....a0 and c0 respectively, their values after the execution will be a6a5 ..... a0c0 and a7 respectively.
JC Y;	Jump to Y if carry flag is set
JMP Y;	Jump to Z
Y: ADD B, # 1	B ← B + 1
JMP Z;	Jump to Z
X:	

133. If the initial value of register A is A_0, the value of register B after the program execution will be

(a) The number of 0 bits in A
(b) The number of 1 bits in A
(c) A_0
(d) 8

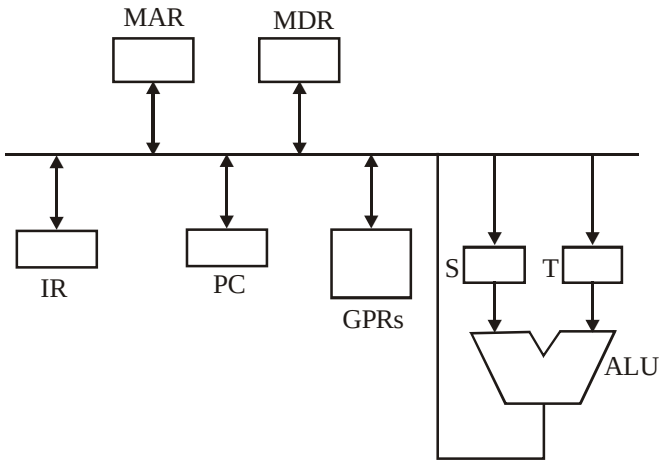
134. Which of the following instructions should be inserted at location X to ensure that the value of register A after program execution is the same as its initial value?

(a) NOP (b) LRC A, # 1

(c) ADD A, # 1 (d) RRC A, # 1

Common data for Q. 135 and Q. 136

Consider the following data path of a CPU



The, ALU, the bus and all the registers in the data path are of identical size. All operations including incrementation of the PC and the GPRs are to be carried out in the ALU. Two clock cycle are needed for memory read operation – the first one for loading address in the MAR and the next one for loading data from the memory but into the MDR.

135. The instruction "add R0, R1" has the register transfer interpretation R0 < = R0 + R1. The minimum number of clock cycles needed for execution cycle of this instruction is

(a) 2 (b) 3

(c) 4 (d) 5

136. The instruction "call Rn, sub" is a two word instruction. Assuming that PC is incremented during the fetch cycle of the first word of the instruction, its register transfer interpretation is

Rn < = PC + 1;

PC < = M [PC]

The minimum number of CPU clock cycles needed during the execution cycle of this instruction is

(a) 2 (b) 3

(c) 4 (d) 5

HINTS & SOLUTIONS

PAST GATE QUESTIONS EXERCISE

1 (a)

Clock cycle	$R_0 = M$ [loc 1]	$R_0 = R_0 + R_0$	$R_2 = R_2 - R_0$
1	IF		
2	RD		
3	EX	IF	
4	MA	RD	
5	WB	EX	
6		MA	IF
7		WB	RD
8			EX
9			MA
10			WB

Thus, total number of clock cycles required = 10

2. (a) As given
Cache size = 32 kbyte = 32 × 2 ^ 10 byte
= 2 ^ 15 byte = 15 bit
Therefore, size of tag = 32 – 15 = 17 bit
Since, 32 byte of block is used by cache, so there actually 1 k blocks and for indexing that we need 10 bit.

3. (a) 1. A[l] = B[j] indexed addressing mode
2. While [*A ++] auto increment
3. int. temp = *x Indirect addressing

4. (c) Following are the memory cycles needed during the execution cycle.
First memory cycle Read value of A to calculate index addresses
Second memory cycle Add R_0 to get the address of first source and read its value
Third memory cycle Read the contents of B
Fourth memory cycle Read the values of the contents of B
Fifth memory cycle Write the answer
→ Total memory cycles needed during the execution cycle of the instruction = 5

5. (a) Given that $R_0 \leftarrow R_0 + R_1$
The clock cycles operate as follows:

Cycle 1	Out : R_1
	In : S
Cycle 2	Out : R_2
	In : T
Cycle 3	Out : S, T
	Add : ALU
	In : R

Therefore, execution cycle is completed in 3 clock cycles.

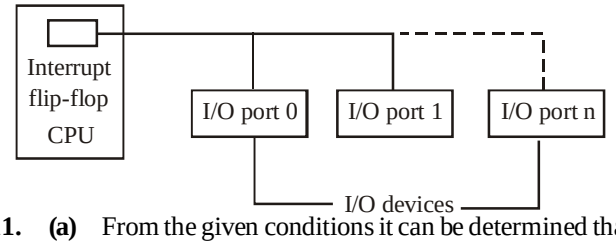
6. (b) The minimum number of CPU clock cycles needed during the execution cycle =4As 1 cycle required to transfer already incremented value of PCand 2 cycle for getting data in MDR1 to load value of MDR in PC.

7. (c) We know that, size is directly proportional to page frame.
So, the RAM is increased. The main memory also increases and thus the page frame size also increases which results in reduction of swapping. Thus, lesser number of page faults occur.
Now, if any replacement rule is applied which doesn't cause blady anomaly always results in reduction of page faults.

8. (b) Let us assume that CPU contains two processes when one process is being executed on the CPU the other one is swapped out and all the data is saved on the disk. Thus the swap space is basically used for saving the process data.

9. (a) Memory mapped I/O means, accessing I/O via general memory access as opposed to specialized IO instructions. An example,
unsigned int volatile const *pMappedAddress const = (unsigned int *)0×100;
So, the programmer can directly access any memory location directly. To prevent such an access, the OS (kernel) will divide the address space into kernel space and user space. An user application can easily access user application. To access kernel space, we need system calls (traps).
So, IO protection is ensured by OS abstraction.

10. (b) In single line interrupt system, vectored interrupts are not possible but multiple interrupting devices are possible as a single line interrupt system consists of a single interrupt system consists of a single interrupt request line and a interrupt grant line in such a system it may be possible that at the same time more than one output devices can request an interrupt, thus in such cases only one request will be granted according to the priority as is depicted by the following figure, but the interrupt is granted to the single request only.



11. (a) From the given conditions it can be determined that we have to set all the other bits to 0 in temp, as the position pos is the only deciding factor in jumping to the label. If the left shift over 1 is done by the pos number then the mask register can have 1 in pos place, which is required. Thus, is the option mask ← 0 × 1 << pos is correct.

12. (b) As given there is no conditional loop for the 80% of 10^9 instruction thus only single cycle is required for them and rest requires 2 extra cycles.
Thus the total time required = Time of one cycle
$= (80/100 \times 10^9 \times 1 + 20/100 \times 10^9 \times 2)$
$= 1/1G \times (80/100 \times 10^9 \times 1 + 20/100 \times 10^9 \times 2) = 1.2s$

13. (d) As given,
Number of banks in main memory = 24 and each is of 2 byte long.
Block size of cache = 64 byte
Due to availability of parallel accessing of the banks, only two accesses are needed to traverse the entire data and the time required for this access comes out to be 92 ns.
At total time = Decoding time + Latency time
$= 24/2 + 80 = 92$ ns
Total latency time for 2 such accesses
$= 2 \times 92 = 184$ ns

14. (c) The number of data cache misses that occur in total are 56 as is clear from the given data.

15. (a) The lines that will be replaced by the new blocks are lines 4 to 11.

16. (d) The memory reference required by the instruction $R_1 \leftarrow M[3000]$ is 1 whereas the memory reference required by $R_2 \leftarrow M[R_3]$ and $M[R_3] \leftarrow R_2$ is 10.
Therefore, total memory reference required = $2 \times 10 + 1$

17. (a) The content in the memory location 2010 doesn't change as the value in register R_1 becomes zero, so the loop BNZ exists and as the address in R_3 becomes 2010. Thus, the content of the location still remains 100.

18. (a) The locations of various operations are given in the following table:

Location	Operation
1000	$R_1 \leftarrow M[3000]$
1001	
1002	$R_2 \leftarrow M[R_3]$
1003	$R_2 \leftarrow R_1 + R_2$
1004	$M[R_3] \leftarrow R_2$
1005	$R_3 \leftarrow R_3 + 1$

Thus, the location of the instruction $INCR_3$ is 1005, if an interrupt occurs, and thus it is pushed on the stack.

19. (b) As given the pipelined processor has four stages i.e., IF, ID, EX, WB,
And we know that number of clock cycles required to ADD and SUB instructions is 1 and by MUL instructions are 3.
In the pipelined processor while one instruction is fetched, the other is either being decoded or executed or some action is being performed. Thus, the number of cycles required by the given set of instructions can be obtained from the following diagram.

Clock cycle	ADD	MUL	SUB
1	IF		
2	ID	IF	
3	EX	ID	IF
4	WB	EX	ID
5		EX	
6		EX	IF
7		WB	EX
8			WB

Thus, total number of clock cycles required are 8.

20. (a) Each address is multiple of 3 as the starting address is 300 and is each instruction consists of 24 bit, i.e., 3 byte.
Thus, in the given options the valid counter will be the one which is the multiple of 3. Out of the options we can see that only 600 satisfies the condition.
Therefore, it is 600.

21. (a) The formula used is
Total disk size is given by = Number of surfaces × Number of tracks × Number of sectors × Capacity of each sector.
Therefore from the given data, we get
Total disk size = $16 \times 128 \times 256 \times 512$ byte
$= 2^8 \times 2^{20} = 2^8$ megabyte = 256 MB
Total number of sectors = $16 \times 128 \times 256$ byte
$= 2^{19}$ byte

22. (d) 64 words require 6 bitsno. of sets = no. of lines / set size = 128/4 = 32 sets which require 5 bitsno. of tag bits = 20-5-6 = 9,5,6

23. (b) The first instruction in the fall through path is executed for all the delayed conditional branch instructions, irrespective of whether the condition evaluates to true or false.

24. (a) I_4 since R1 is only read in I_4 and its value will remain as it is while BRANCH is executed and after BRANCH get executed , since in BRANCH R1 is only read.

25. (b)

Tag	Set	Block
17	11	4

From the above figure and given conditions the total number of tags comes out to be = $17 \times 2 \times 1024 = 32$ bit

26. (b) The array element ARR [4] [0] has the same cache index as ARR [0] [0] since it is given that page size is of 4 kbyte and 1 row contains 1024 elements, i.e., 2^{10} locations.

27. (b) We know that hit ratio is given as
Number of hits/ Number of hits + Number of rows
= 4/16 = 25%

28. (d) When an RFE (Return From Exception) instruction is executed; no exception is allowed to occur during that time and also it must be a trap instruction and also a privileged one.

Thus all the three statements are true as far as RFE is concerned.

29. (c) For incrementing the data, the auto-increment addressing mode is used which purely depends on the size of the data. For example:
Regs [R_1] ← Regs [R_1] + Mem [Regs [R_2]
Regs [R_2] ← Regs [R_2] + d

30. (c) The seek latency is not linearly proportional to seek distance due to the higher capacity of tracks on the periphery of the latter. The higher capacity of the tracks is responsible for the presence of this certain amount of time is required for this cells to reach the read-write head so that data transfer can take place.

31. (d) The interrupt register is checked after finishing the execution of the current instruction. At this time, a CPU generally handles an interrupt by the execution of an interrupt service routine.

32. (c) As given, basic RAM is 32 k × 1 and we have to design a RAM of 256 k × 8.
Therefore, number of chips required
= 256 k × 8/(32 k × 1)
= 245 × 1024 × 8/32 × 1024 × 1)
(Multiplying and dividing by 1024)
= 64 = 8 × 8
Means, 64 = 8 parallel lines × 8 serial RAM chips.

33. (c) We have to find the sector number of the address <400, 16, 29>
Therefore, 400 * 2 * 10 * 63 + 16 * 63 + 29
= 505037 sector

34. (c) <0, 16, 13> this address corresponds to a sector number which is given by 16 * 63 + 31 = 1039

35. (d)

Set 0	48 4 32 8 26 92
Set 1	1 133 73 129
Set 2	
Set 3	255 155 3 159 63

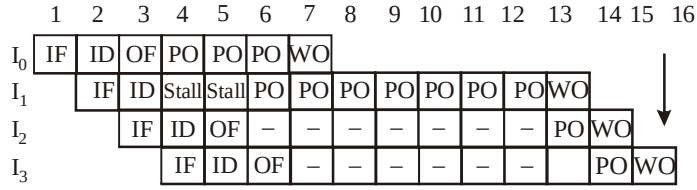
0 mod 4 = 0 (set 0)
255 mod 4 = 3 (set 3)
Like this the sets in the above table are determined for the other values.

36. (d) Number of cycle for executing loop i=1 is 15 so, total no. of cycle is 2*15(loop runs 2 time), if we calculate no. of pipeline continuous no option will match, so here we have to take loop independently.

37. (c) As already given in question,
Memory access time for L_1 = 2 ns
Memory access time for L_2 = 20 ns
Now the required time of transfer = 20 + 2 = 22 ns

38. (a) As given memory access time for main memory
= 200 ns
Memory access time L_2 = 2 ns
Memory access time L_2 = 20 ns
Total access time = Block transfer time from main memory to L_2 cache + Access time of L_2 + Acces time of L_1
Now, the required time of transfer
= 200 + 20 + 2 = 222 ns

39. (b) As per the given, the instructions are arranged accordingly to their meanings. We get the following:



Here, we can see that the last operation (Write Operand) comes at the 15th clock cycle so it takes 15 clock cycles to execute given sequence of instructions.

40. (d) One chip contains 10^6 bits.to build 4MB capacity MM 32 chips will be required there are 1K × 1K cells in each chip i.e. 10^6 cellshence time taken to refresh a single chip 10^6 × 100 nsthere are 32 such chips hence 32 × 10^6 × 100 ns time will be takeni.e. 3200 × 2^20ns

41. (a) Number of clock cycles required by susing load-store approach = 2 + 500 × 7 = 3502 and that of by using DMA = 20 + 500 × 2 = 1020
Required speed up = 3502 / 1020 = 3.4

42. (d) Direct mapped cache => number of sets = 8KB/32B = 2565 bits for offset within a block & 8 bits for identifying set So remaining 32-8-5=19 bits for tag.1 tag/32B block it includes 2 more bits. So 21 bits/32 B block Number of such blocks = 256 So total tag mem = 256 × 21 = 5376

43. (b) Speed up = $\frac{(5+6+11+8)\times 1}{(11+1)} = \frac{30}{12} = 2.5$

44. (d) Higher priority interrupt levels are assigned to requests which, if delayed or interrupted, could have serious consequences. Devices with high speed transfer such as magnetic disks are given high priority and slow diviser such as keyboard receive low priority. Mouse pointer moments are more frequent then keyboard ticks. So its obvious that its data transfer rate is higher than keyboard. Delaying a CPU temperature sensor could have serious consequences, overheat can damage CPU circuitry. From the above information we can conclude

that priorities are - CPU temperature sensor > Hard disk > Mouse > Keyboard.

45. (d) The addressing mode will be base index addressing. Here, 20 will work as base and content of R_2 will be index. R_2 is base regester here and indexing is done by adding 20 to the contents of R_2.

46. (b) Effective access time = p × page fault service time + (1 – p) × memory access time

$$\text{where p = page fault rate} = \frac{1}{10^6} = 0.000001$$

memory access time = 20 ns
Page fault service time = 10 ms = 10000000
$\Rightarrow 0.000001 \times 10000000\,\text{ns} + 0.999999 \times 20\,\text{ns} = 29.9998$
≈ 30 ns

47. (d) The normal size of ROM is $n \times 2^n$
$\therefore$ Now, we are multiplying two n-bit numbers.
So, the resultant has 2n bit.
Hence, the size of the ROM is $2n \times 2^{2n}$
In the question, n = 4
Hence, $\Rightarrow 2 \times 4 \times 2^{2\times4}$
$\Rightarrow 8 \times 2^8 \Rightarrow 2^3 \times 2^8$
$\Rightarrow 2 \times 2^{10} \Rightarrow 2$ k bit
OR
As there is two independent 4 bit inputs, so you need 2 × 4 = 8 address lines A 4 bit × 4 bit multiplication has 8 bit result, you need 8 data lines.
$\therefore$ Rom needs to be at least 256 × 8 = 2 k bit.

48. (c) Pipelining of the register renaming logic can help avoid restricting the processor clock frequency.

49. (b) Considering the following case such that the cache blocks are arranged as follows

0	3	6	9	12
1	4	7	10	13
2	5	8	11	14

The cache is divided into v = 3 sets and each of which consists of k = 5 lines.
e.g. suppose we need to find the block 9, then
Option (a) :
(9 mod 3) * 5 to (9 mod 3) * 5 + 4 = 0 to 4
Option (b) :
(9 mod 3) to (9 mod 3) + 4 = 0 to 4
Option (c) :
(9 mod 5) + (9 mod 5) + 2 = 4 to 6
Option (d) :
(9 mod 5) * 3 to (9 mod 5) * 3 + 2 = 12 to 14
Now check for option (a), (b), (c) and (d).
Check for block 11.
Option (a) :
(11 mod 3) * 5 to (11 mod 3) * 5 + 4 = 10 to 14
Option (b) :
(11 mod 3) to (11 mod 3) + 4 = 2 to 6
Option (c) :
(11 mod 5) to (11 mod 5) + 2 = 6 to 8
Clearly, option (a) gives correct answer.

50. (d) The following sequence of micro-operations

MBR ← PC
MAR ← X
PC ← Y
Memory ← MBR.

Analysis

1. First micro operation stores the value of PC into Memory Base Register (MBR).
2. Second micro operation stores the value of X into Memory Address Resister (MAR).
3. Third micro operation stores value of Y into PC.
4. Fourth micro operation stores value of MBR to memory.

So before execution of these instructions PC holds the value of next instruction to be executed. We first stores the value of PC to MBR and then through MBR to memory i.e., We are saving the value of PC in memory and then load PC with a new value. This can be done only in two types. Operations Conditional branch and interrupt service.
As we are not checking here for any conditions.
So, it is an Initiation of interrupt service.

51. (d) Starting disk location of the file is <1200, 9,40 >.
Since starting number of sectors left on 9th surface is = 24. So, on 9th surface total storage of "12288" B is possible. Now, a part from 9th surface, on cylinder number 1200 only 6 surface is left,
Storage possibilities on these 6 surfaces are
$= 6 \times 2^6 * 2^9$ storage on each sector
number of sectors on each surface
=196608 B
The total on cylinder no 1200, storage possible
= 196608 + 12288 = 208896 B.
Since file size is 42797 KB and out of which 208896 B are stored on cylinder, no 1200. So we are left only with 43615232 B.
Since in 1 cylinder, storage possible is
$= 2^4 * 2^6 * 2^9$ B = 524288 B.

$$\text{So, we need about} = \frac{43615232\,\text{B}}{524288\,\text{B}} = 83.189 \text{ more}$$

cylinders.
Hence, we will need 1284th cylinder to completely store the files since after 1283rd cylinder we will be left with data which will 189.

52. (b)

53. (b) Instruction pipeline with five stages without any

branch predicition: Delays for FI, DI, FO.EI and WO are 5,7,10,8,6 ns respectively.

Explanation The maximum time taken by any stage is 10 ns and additional 1ns is required for delay of buffer.

∴ The total time for an instruction to pass from one stage to another in 11 ns.

The instructions are executed in the following order $l_1, l_2, l_3, l_4, l_9, l_{10}, l_{11}, l_{12}$

Execution with Time

Now when l_4 is in its execution stage we detect the branch and when l_4 is in WO stage we fetch I_9 so time for execution of instructions from l_9 to l_{12} is = 11* 5 +(4 – 1) * 11= 88 ns.

But we save 11 ns when fetching l_9 i.e., l_9 requires only 44 ns additional instead of 55ns because time for fetching l_9 can be overlap with WO of l_4.

∴ Total Time is = 88 + 88 – 11 = 165 ns

54. (b) Assuming that R_1 and R_2 holds the value of a and b respectively. The machine code is equivalent to

$R_2 \leftarrow R_1 + R_2$ (c = a + b) (R_2 holds c)
$R_3 \leftarrow R_2 * R_1$ (d = c * a)
$R_4 \leftarrow R_2 + R_1$ (e = c + a)
$R_2 \leftarrow R_2 * R_2$ (x = c * c)

(Removing c from R_2 as c is not used after this statements R_2 holds x)

If ($R_2 > R_1$) { {If (x > a)}
If ($R_2 > R_1$)
{
$R_2 \leftarrow R_1 \times R_1$
$R_2 \leftarrow R_1 * R_1$ (y = a* a)
}
else {
$R_3 \leftarrow R_3 * R_3$ {d =d *d)
$R_4 \leftarrow R_4 * R_4$ (e = e * e)
}

We have used R_1, R_2, R_3,R_4 registers

So answer is (b) 4.

55. (b) In this question we have to minimize spills to memory (writer). As variables d,e are only used in else part. So we can move d,e instructions inside else. The code we get is

c = a + b; Assumption-Register R_1 and R_2.
x = c * c; Already holds a and b respectively,
if (x > a)
{ We can first calculate the number of
y = a * a memory spill. The machine code is equivalent to
else {
d = c * a; $R_2 \leftarrow R_1 + R_2$ (c = a + b)
d = d*d; write R_2 to memory (So memory has value of c)
e = e * e if ($R_2 > R_1$) x = c * c
} { (x > a)
$R_2 \leftarrow R_1 + R_1$ (y – a * a)
}
else {
$R_2 \leftarrow$ (read value of c from memory) R_1 (d = c * a)
$R_2 \leftarrow R_2 * R_2$ (d = d * d)
$R_2 \leftarrow$ (read value of c from memonry) + R_1 (e = c + a)
$R_2 \leftarrow R_2 * R_2$ (e = e * e)
}

⇒ There has been only 1 memory write observed in the give code.

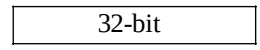
So the answer is 1.

56. (b)

57. (c)

58. 16383

(i) Given is 32-bit architecture machine with 1 word long instruction; so, the instruction size is 32 bit.



(Instruction)

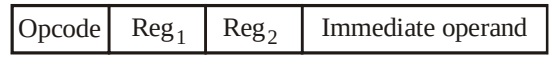
(ii) As the machine need to support 45 instructions, the number of bits required in the "opcode field" is 6 bits, because,

5-bits: $2^5 = 32$ (Not sufficient)
6-bits: $2^6 = 64$ (Sufficient)

(iii) As the machine has 64 registers, so to uniquely identify a register out of those 64 registers, we need 6-bits. (using the similar argument as above)

As each instruction has 2 register operands, so 6 + 6 = 12 bits are needed for them.

So, finally instruction will took like as follows:



(Total = 32 bits)

(iv) As 14-bits are available for immediate operand, the max value will be $2^{14} = 16384$.

However one value will be used for "Zero" also as the operand is an "unsigned integer". So, finally, the maximum value that the immediate operand can take $= 2^{14} – 1 = 16383$.

59. 4 to 4

60. (A) $\frac{n}{N}$

61. 20

Physical address size = 32 bit
Cache size = 16k bytes
$= 2^{14}$ bytes

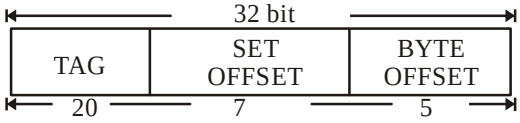
block size = 8 words
= 8 × 4 byte
= 32 byte

$$\text{No. of blocks} = \frac{2^{14}}{2^5} = 2^9$$

block offset = 9 bits

$$\text{No. of sets} = \frac{2^9}{4} = \frac{2^9}{2^2} = 2^7$$

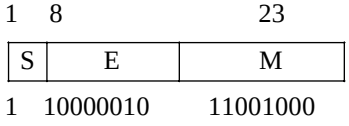
set offset = 7 bits
Byte offset = 8 × 4 bytes
= 32 bytes
$= 2^5$
$\Rightarrow$ 5 bits
TAG = 32 – (7 + 5) = 20 bits



62. (d) A smaller cache block size accomodate more number of blocks, it improves hit ratio of cache, so the miss penalty is covered.

63. (d) If the associativity of a processor cache is doubled while keeping the capacity and cache block size unchanged, the width of processor to main data bus is not affected.

64. (a) $14.25 = 1110.010$
$= 1.110010 \times 2^3$
$\therefore m = 110010$



(= C1640000)

E = 127 + 3 = 130
= 10000010

65. **10000**
Each write request, the basic occupied for 100 ns
Storing the data requires 100 ns.
In 100 ns – 1 store

$$\frac{100}{10^6}\ \text{ms} \rightarrow 1\ \text{store}$$

$$1\ \text{ms} = \frac{10^6}{100}\ \text{stores}$$

= 10.000 stores

66. (c) $\text{Frequency} \propto \frac{1}{\text{clock period}}$

Clock period = maximum stage delay + overhead

P_1 : CP = Max (1, 2, 2, 1) = 2 ns
P_2 : CP = Max (1, 1.5, 1.5, 1.5) = 1.5 ns
P_3 : CP = Max (0.5, 1, 1, 0.6, 1) = 1 ns
P_4 : CP = Max (0.5, 0.5, 1, 1, 1, 1) = 1.1 ns
$\because$ CP of P_3 is less, it has highest frequency

$$\text{Frequency 1 } P_3 = \frac{1}{1\text{ns}} = 1\text{GHz}$$

67. **1.54**

	No. of stages	Stall cycle	Stall frequency	Clock period	Avg. access time
Old design	5	5	20%	2.2ns	P
New Design	8	5	20%	1 ns	Q

$$P = \left[80\%(1\text{clock}) + 20\%\left(\underset{\text{completion}}{1} + \underset{\text{stall clock}}{2}\right)\right] \times T_{c-p}$$

P = (.8 + .6) × 2.2ns = 3.08ns

$$Q = \left[80\%(1\text{clock}) + 20\%\left(\underset{\text{completion}}{1} + \underset{\text{stall clock}}{5}\right)\right] \times T_{c-p}$$

P = (.8 + .12) × 1ns = 2ns

$$\text{So the value of } \frac{P}{Q} = \frac{3.08\text{ns}}{2\text{ns}} = 1.54$$

68. **1.68**
Total instructions = 100 instruction fetch operation + 60 memory operant read operation + 40 operant write operation
Therefore, total of 200 instructions
Time taken for fetching 100 instructions (equivalent to read)
= 90 × 1ns + 10 × 5ns = 140ns
Memory operand Read operations = 90%(60) × 1ns + 10% (60) × 5ns
= 54ns + 30ns = 84ms
Memory operands write operation time = 90%(40) × 2ns + 10%(40) × 10ns
= 72ns + 40ns = 112ns
Total time taken for executing 200 instructions = 140 + 84 + 112 = 336ns

$$\therefore \text{ Average memory access time} = \frac{336}{200}\text{ns} = 1.68\text{ns}$$

PRACTICE EXERCISE

1. (c) **2. (c)** **3. (c)**

4. (a) Here [X] command will require 3 address spaces because the instruction contains address 00FF or 0701 which is 16-bit data. Thus the address will be as

Adress

0100 LXI SP, 00FF

0103 LXI H, 0701

0106 MVI A, 20 H

0199.....

Here, MVI A, 20 means 20 is to be stored in accumulator. Thus A ← 20 H

Thus the content of accumulator will be 20 H

5. (c) MVI05 as 05 is data that is directly given in the instruction.

6. (b) 2 cycle average access = (1 cycle for cache) + (1 - hit rate)(5 cycles stall) => hit rate = 80%

7. (b) In direct mode address is given in the instruction which is 400 means 0-3.

In immediate addressing mode data is given in the instruction that is data is 400 which is stored at 301.

8. (b) **9. (b)** **10. (a)** **11. (a)** **12. (d)**

13. (a) Locality implies that the probability of accessing a location decreases as the time since the last access increases. By choosing to replace locations that haven't been used for the longest time, the least-recently-used replacement strategy should, in theory, be replacing locations that have the lowest probability of being accessed in the future.

14. (c) **15. (c)** **16. (c)** **17. (a)** **18. (a)** **19. (b)**

20. (b) **21. (d)** **22. (b)** **23. (b)** **24. (b)** **25. (d)**

26. (b) **27. (a)**

28. (a) Prefetch target instruction strategy

29. (d) Speed up factor of a pipeline

= Number of stages in pipelines = n

30. (d) Total word in a set = $64 \times 4 = 2^6 \times 2^2 = 2^8$

$$\text{Number of sets in cache} = \frac{4 \times 2^{10}}{2^8} = 4 \times 2^2 = 16 = 2^4$$

Thus the cache will have 16 sets which will requires 4-bit.

Total memory size = 4 k = $4 \times 2^{10} = 2^{12}$

Available address bits = 12

Required bits for set = 4

Required bits for block = 2 (as 4 block are in 1 set)

Thus required bits for word = 12 – (4 + 2) = 6

31. (b) The combination of input data will be 2^8. The combination of two 8-bit data will be

$2^8 \times 2^8 = 2^{16}$

Thus, 2^{16} bit memory location will be required to store all combination of words.

2^{16} bits = 2^6 k = 64 k memory

and 2^{16} bit will be required to store the result of each multiplication thus total memory will be 64k × 16 bit

32. (c) The address <400, 16, 29> corresponds to sector number

400 × 2 × 10 × 63 + 16 × 63 + 29 = 50 50 37

33. (b) The Register Transfer Language (RTL) is

$R_3 \leftarrow R_3 + M[R_1 + R_2]$

In the given statement R_1 is the base of an array and R_2 represents an index so indexed addressing mode is suitable for given RTL.

34. (b) The instruction format is as follow

Mode	opcode	address	address designer
1 bit	3 bit	12 bit	1

Now Mode = 1 if indirect addressing

Mode = 0 if direct addressing

35. (d) CISC – Complex instruction set computer

RISC – Reduced instruction set computer

36. (b)

37. (b) (i) Regs [R_4] ← Regs [R_4] + Regs [R_3]

Here contents of Regs [R_4] and [R_3] are added and placed into Register [R_4] which is a register operation.

(ii) Regs [R_4] ← Regs [R_4] + 3

Here 3 is data instead of address

Hence, it is a immediate mode.

Hence (b) is correct option

38. (c) In pipelining, a number of functional units are employed in sequence to perform a single computation.

39. (d) 2^width of the address bus x the data bus = Maximum amount of addressable memory.

32 bit system with a 32-bit address bus width and a 32-bit data bus width. 2^32 x 32 = 13743895350Bytes = 128GB, hence a 32 bit system can only address 4GB of memory.

40. (b) **41. (c)** **42. (c)** **43. (d)** **44. (c)**

45. (d) **46. (c)**

47. (a) Increased block size means that more words are fetched when filling a cache line after a miss on a

particular location. If this leads to increased performance, then the nearby words in the block must have been accessed by the program later on, ie, the program is exhibiting locality.

48. (a) Considering only the data accesses, the program performs 1,000,000 reads and 1,000,000 writes. Since the cache has 4-word blocks, each miss brings 4 words of the array into the cache. So accesses to the next 3 array locations won't cause a miss. Since the cache is write-back, writes happen directly into the cache without causing any memory accesses until the word is replaced. So altogether there are 250 misses (caused by a read of A[0], A[4], A[8], ...), for a miss rate of 250/2,000,000 = 0.0125%

49. (d)

50. (d) There are 64 words in a block. So the 4k cache has (4 × 1024)/64 = 64 blocks. Since 7 set has 4 blocks, there are 16 sets. 16 sets needs 4 bits for representation. In a set there are 4 blocks, which needs 2 bits. Each block has 64 words. So the word field has 6 bits.

51. (b)

52. (c) The operation code specifies one of eight input out put instructions. Start input/output, start input/ output fast release, test input/output, clear input/ output, halt input/output, halt device, test channel and store channel identification.

53. (d) 3 address institruction

Two operand locations and a result location are explicitly contained in the instruction word.

e.g., Y = A – B

2 address instruction

One of the addresses is used to specify both an operand and the result location.

e.g., Y = Y + X

1 address instruction

Two addresses are implied in the instruction and accumulator based operations.

e.g., ACC = ACC + X

0 address instructions

They are applicable to a slpecial memory organization called a stack. It interact with a stack using push and pop operations. All addresses are implied as in register based operations.

T = Tap(T – 1)

54. (d) The secondary cache can be considerably slower, but it shold be much larger to ensure a high hit rate if speed is less critical, because it only affects the miss penalty of the primary cache. A workstation computer may include a primary cache with the capacity of tens of kilobytes and a secondary cache of several megabytes.

$t_{avg} = h_1c_1 + (1 - h_1)h_2c_2 + (1 - h_1)(1 - h_2)M$

The number of misses in the secondary cache, given by the term $(1 - h_1)(1 - h_2)$.

55. (d) **56. (a)**

57. (b) The cache has 128-byte lines, which is 32 words. Using the specified memory timings, the cache spends 5 ns detecting that a miss has occurred and 100 ns + 31(10 ns) = 410 ns fetching a line from the main memory into the cache. The cache then reexecutes the operation that caused the miss, taking 5 ns. This gives a cache miss time of 420 ns.

58. (b) $-1 \times 2^{\circ} + 1 \times 2^3 - 1 \times 2^6 = -1 + 8 - 64 = -57$

59. (d) Daisy chained arbitration scheme requires 3 lines irrespective of the number of devices. A pair of REQ/ GRANT lines are required in the case of independent request type arbitration and so for N devices 2N lines.

60. (d) In index Addressing mode, the effective address of the operand is generated by adding an index values to the address given in the instruction. The index value is held in a register called index register.

$$EA = \underset{\text{modifier}}{A} + \underset{\text{contents of index register}}{A}$$

61. (c) **62. (d)** **63. (a)** **64. (d)**

65. (b) Speed up

$$= \frac{1}{\left(1 - \%\text{ of time cache can be used}\right) + \left[\frac{\%\text{ of time cache can be used}}{\text{speed up using cache}}\right]}$$

$$= \frac{1}{(1-0.7)+\frac{0.7}{10}} = \frac{1}{0.3+0.07} = \frac{1}{0.37} = 2.7$$

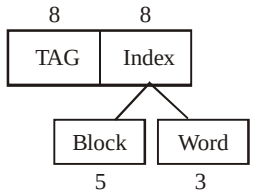
Hence, we obtain a speed up from the cache of about 2.7 times

66. (b) **67. (d)**

68. (d) The performance of a pipelined processor depends upon delays of different stage and its hardware resources also it depends upon consecutive instructions format.

69. (d) Memory address: 16 bits

Number of words in cache = 256 = 2^8



70. **(b)**

71. **(c)** Initially $R_0 \leftarrow X, R_1 \leftarrow Y, R_2 \leftarrow Z$

The sequence of operations to compute XY + XYZ + YZ is as follows

Operation	Meaning	No. of clock cycles
MUL R_0, R_1, R_3	$R_3 \leftarrow XY$	1
MUL R_1, R_2, R_5	$R_5 \leftarrow YZ$	1
MUL R_2, R_3, R_4	$R_4 \leftarrow XYZ$	1
ADD R_3, R_4, R_6	$R_6 \leftarrow XY + XYZ$	1
ADD R_5, R_6, R_7	$R_7 \leftarrow XY + XYZ + YZ$	2
		Total 6 clock cycles

72. **(d)** Average access time = Hit ratio × Cache access time + Miss ratio × Memory access time

Here, memory access time = 1.5 μs i.e. 1500 ns

So, average across time = 0.98 × 12 ns + 0.02 × 1500ns

= 11.76 ns + 30 ns = 41.76 ns

73. **(b)**

74. **(c)** The diagram shows, extended addressing mode. In this, the effective memory address directly specified and it is used by some of the processor and address specified is 16 bit address.

75. **(b)**

76. **(c)** There are 7 pushes and pops for a cost of 35 bytes plus 3 arithmetic instructions for a total of 38 bytes storage.

After instruction		Stack contains	New variables
push	b	b	
push	x	b, x	
add		b + x	
pop	c		c = b + x
push	c	b + x	c = b + x
push	y	b + x, y	c = b + x
add		b + x + y	c = b + x
push	c	b + x + y, b + x	c = b + x
sub		y	c = b + x
pop	z		c = b + x, z = y

77. **(b)**

Initially	VC	Z	1100	1100
	00	0	1000	1111
After addition	11	0	10101	1011

V-bit is set to 1 due to arithmetic overflow.

C-bit is set to 1 because most significant digits generates a carry.

Z-bit is set to 0 because the result of addition is not zero.

78. **(a)**

79. **(a)** A ⇒ Two address instruction

B ⇒ One address instruction

C ⇒ Zero address instruction

80. **(a)** Pipelined operation is interrupted whenever two operations being attempted in parallel need the same hardware component, such a conflict can occur if the execution phase of an instruction requires access to the main memory, which also contains the next instruction that should be fetched at the same time.

81. **(c)** Average instruction execution time

= 1 ns × ((40% + 20%) × 4 + 40% ×5) = 4.4 ns

Speed up from pipeline

= Average instruction time unpip lined/Average instruction time pipelined

$$= \frac{4.4}{1.2} = 3.7$$

82. **(c)** **Control command:** A control command is issued to activate the peripheral and to inform it what to do.

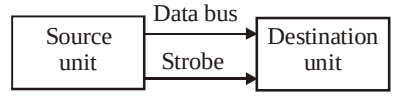
Status: A status comman is used to test various status conditions in the interface and the peripheral.

Data output command causes the interface to respond by transferring data from the bus into one of its register.

83. **(c)** A synchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted.

Two ways of achieving this

1. By means of a strobe pulse supplied by one of the units to indicate to the other unit when the transfer has to occur.



2. To accompany each data item being transferred with a control signal that indicates the presence of data in the bus. The unit receiving the data item responds with another control signal to acknowledge receipt of the data. This type of agreement between two independent units is referred to as handshaking.

84. **(a)** Rounding to plus and minus infinity are useful in implementing a technique known as interval arithmetic. Interval arithmetic provides an efficient method for monitoring and controlling errors in floating point computations by producting two values for each result. The two values correspond to the lower and upper endpoints of an interval that contains the true result.

The width of the interval, which is the difference between the upper and lower endpoints, indicates the accuracy of the result of the endpoints of an interval are not representable then the interval endpoints are rounded down and up respectively.

85. (d) Effective access time = (hit ratio) × (memory access time + search time in associative registers) + (fail ratio) × (search time in associative registers + 2 × memory access time) = (0.90 × 150) + (0.10 × 250) = 160 nanoseconds.

Now percentage slowdown = 160 – 100 = 60

86. (b) The first format defines 2^6 patterns, among which 3 are not used for op-codes. The second format occurs for these 3 cases, and each case corresponds to 2^6 patterns.

So, the total number of op-codes = $2^6 - 3 + 3 \times 2^6 = 61 + 192 = 253$.

87. (d) **88. (d)** **89. (c)**

90. (b) DMA Clock is 2 MHz ⇒ Each DMA Clock state is 0.5 μS

Each DMA Cycle has 6 Clock States ⇒ Each DMA cycle is 3 μS

In Cycle Steaing 1 CPU and 1 DMA Cycles run alternately and the CPU Cycle takes 2 μS.

Therefore, every 3 + 2 = 5 μS, 1 Byte is transferred by DMA device.

Data Transfer Rate = 1000000/5 × 1 Byte

= 200 Kbytes/Sec.

91. (b) Next Address Field = 7 Bits

ALU function field is of 4 Bits (to select possible 1 of 13 ALU fns)

Register-in and Register-out fields of 3 Bits each

Shifter field of 2 bits to select 1 out of three possibilities

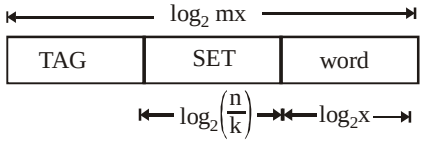
Auxiliary field of 4 Bits

Total Bits in μ-instruction = 7 + 4 + 3 + 3 + 2 + 4 = 23

92. (b) **93. (a)**

94. (a) Number of sets = $\frac{n}{k}$

Let there are x words per block in main memory.



$$\text{Number of TAG bits} = \log_2(mx) - \left(\log_2\frac{n}{k} + \log_2 x\right)$$

$$= \log_2(mx) - \log\frac{nx}{k} = \log_2\frac{mk}{n}$$

95. (b)

96. (b) We are asked minimum performance gain , So consider for transferring minimum amount of data which is 1 Byte. For program controlled mode, CPU will be busy for 1/(10 * 2 ^ 10) = 97.66 microsec.

For interrupt mode extra overhead = 4microsec. So minimum performance gain is 25 approx.

97. (b) **98. (b)** **99. (b)**

100. (c) Minimum cycle time =latency of longest module in data path plus pipeline latch time =7ns + 1 ns=8 ns.

assuming that thereis no limit on the number of pipeline stages. We need to know how many pipeline stages the processor requires to operate at a cycle time of 8ns. We can group any set of adajacent modules with total latencies of 7 ns or less into a single stage. Doing this gives 5 pipeline stages.

5 stages * 8 ns cycle time = 40 ns latency

101. 165.420 microsecond

102. 4.76

$$S = \frac{nt_p}{(k+n-1)t_p} = \frac{100\times 50}{(6+100-1)\times 10} = \frac{500}{105}$$

103. the current generation of chips (DDR SDRAM) has a refresh time of 64 ms and 8,192 rows, so the refresh cycle interval is 7.8 μs

104. 3

105. 21

A virtual page with 32-bit addresses between 0xffff e800 to 0xffff efff

- Virtual Page size = (0xffff efff - 0xffff e800 + 1) = 2 × 1024 B = 2 kB
- Number of offset bits = log2 (2 × 1024) = log2(211)
- Number of VPN bits = 32 - 11

106. 5.3

Let M = main memory access time

C = cache memory access time $= \frac{M}{10}$

118

Total access time using cache = 0.9 + 0.1 M

$= 0.9 \frac{M}{10} + 0.1\ M = 0.19\ M$

$\text{Speed up} = \frac{M}{0.19\ M}$

107. 28

$\text{Average cost} = \frac{C_1S_1 + C_2C_2}{S_1 + S_2}$

$= \frac{1024 \times 100 + 4096 \times 10}{1024 + 4096} = \frac{143360}{5120}$

108. 00111000

Register R contains : 10011100

After arithmetic shift right register R contains 11001110

After arithmetic shift left register R contains 00111000

109. 64

Memory capacity is 4096 × 16 Each chip is 128 × 8

No. of chips which is 128 x 8 of 4096 x 16 memory capacity

=4096 * 16/ (128 *8) =64

110. 16

Number of RAM chips required = 2048 / 128 =16

111. 11

2048 =2(pow, 11) . Hence 11 lines of the address bus must be used to access 2048 bytes of memory.

112. 6 microseconds

At 10,000 r/min, it takes 6ms for a complete

rotation of the disk. On average, the read/write head will have to wait for half rotation before the needed sector reaches it, SC) the average rotational latency will be 3ms. Since there are 1024 sectors on the track, the transfer time will he equal to the rotation time of the disk divided by 1024, or approximately 6 microseconds.

113. 6

2(pow,6)= 64, so the 6 bits of address determine an address's byte within a cache line.

114. 1024

The number of lines in cacheis simply the capacity divided by the line length, so the cache has 1024 lines with 32-byte lines, 512 lines with 64-byte lines, and 256 lines with 128 byte lines.

115. 64

With 128-byte lines, the cache contains a total of 128 lines.The number of sets in the cache is the number of lines divided by the associativity so cache has 64 sets if it is 2-way set association, 32 sets if 4-way set associative, and 16 set if 8-way set-associative.

116. 34

Using the formula,

The average access time =(THitX H1t) + (TmissXmiss)The average access time is (12 ns x 0.75) + (100 ns x 0.25) = 34 ns

117. 96.2

The average access time

10ns = (8ns × hit rate) + 60 ns × (1–(hit rate)),

(The hit and miss rates at a given level should sum to 100 percent). Solving for hit rate, we get required hit rate of 96.2%.

118. 4

No of RAM chips = 2048/ 256 =8 .

No of ROM chips = 4096/ 1024=4 (note that rest of the info is given to confuse and is not required)

119. 1.4

CPI = (1 inst-per-cycle) + (0.05)(8 cycles/miss) = 1.4

120. (c) 121. (c) 122. (c) 123. (a) 124. (c)

125. (a)

Tag(6 bits)	Set(6 bits)	Word (4 bits)

126. (c) 357 A, whose 16 bits are 0011010101111010, the first 6 bits of which determine the "tag" value. The decimal value of 001101 is 13.

127. (b)

Instructions		No. of clock cycles
AC: = M(X)	LOAD A	1
AC: = AC + M(Y)	ADD Y	1
M(Z): = AC	STORE Z	1
		Total = 3

128. (c)

Instructions		No. of clock cycles
DR: = AC	MOVE DR, AC	1
AC: = AC – DR	SUB	1
		Total = 2

129. (a) Average Memory Access time

= Hit Time L_1 + Miss rate L_1 × (Hit time L_2 + Miss rate L_2 × Miss Penalty L_2) = 1 + 4% (10 + 50% × 100)

$\left[\text{Miss rate } L_1 = \frac{40}{1000} \times 100 = 4\%\right]$

$= 1 + 4\% \times 60 \left[\text{Miss rate } L_1 = \frac{40}{1000} \times 100 = 4\%\right]$

= 3.4 clock cycles

130. (c) Average Memory Stall per instruction

= Misses per instruction L_1 × Hit time L_2

+ Misses per instruction L_2 × Miss penalty L_2

= (60/1000) × 10 + (30/1000) × 100 = 3.6 clock cycles.

131. (b) Average instruction execution time

= Clock cycles × Average Cp_1

= 1[(40% + 20%) × 4 + 40% × 5)] ns

= 1 × 4.4ns = 4.4 ns

132. (c) Average instruction time pipelined = 1 + 0.2 = 1.2 ns

Speedup from pipelined = $\frac{4.4}{1.2}$ = 3.7 ns

133. (b) **134. (b)** **135. (b)** **136. (b)**