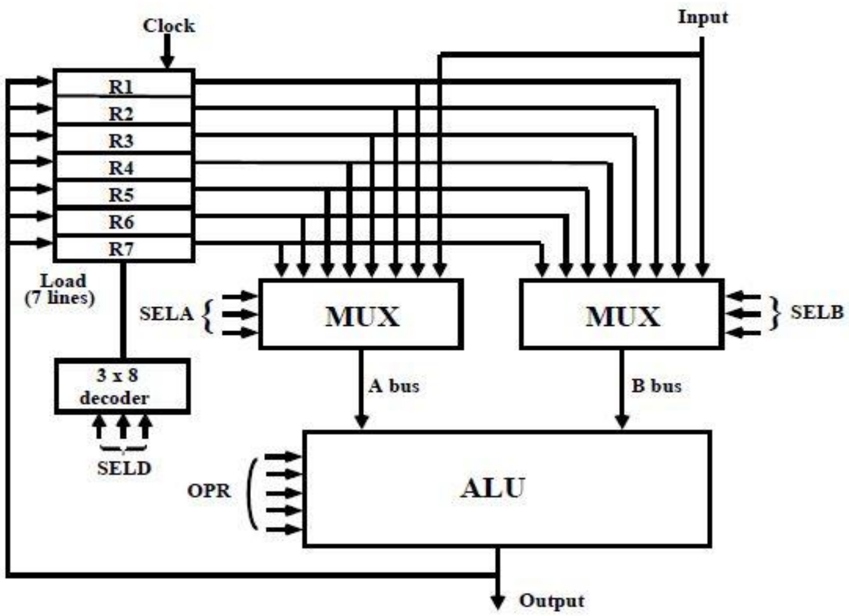
1.Explain the Register set with common ALU. OR Explain the general register organization

- A bus Organization for seven CPU registers is shown in figure. Output of each register is connected to two multiplexer to form the two buses A and B. The selection line in each multiplexer select one register or the input data for a particular bus. The A and B buses forms the input to a common arithmetic logic unit (ALU).The operation selected in the ALU determines the arithmetic or logic microoperation that is to be performed. The result of the microoperation is available for the output data and also goes into the input of all register. The register that receives the information for the output bus is selected by decoder. The decoder activates one of the register load inputs, thus providing a transfer path between the data in the output bus and the input of the selected destination register.



2. Define control word. Explain the procedure for determining control word for specific operation.

- Control word is defined as a word whose individual bits represent the various control signals. Therefore each of the control steps in the control sequence of an instruction defines a unique combination of 0s and 1s in the Control Word. In General register organization there are 14 binary section inputs in the unit and their combined value specifies the control word.
- The 14 bit control is defined in figure below.
  It consists four fields
  1) SELA: 3 bit Selects a source register for the A input of the ALU
  2) SELB: 3 bit selects a source register for the B input of the CPU
  3) SELD: 3bit selects the destination register using the decoder and its seven load output.
  4)The Five bit of OPR select one of the operation in the ALU

The 14 bit control world when applied to the selection input specifies a particular microoperation.

SELA(3)	SELB(3)	SELD(3)	OPR(5)

The three bits each of SELA , SELB , SELD has the following operation:

Binary code	SElA	SELB	SELD
000	INPUT	INPUT	NONE
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
…	..;		
111	R7	R7	R7

When SELA or SELB =000 the corresponding multiplexer selects external data. When SELD=000 no destination register is selected but the content of the output bus are available in the external output.

OPR select	Operation	SYMBOL
00000	Transfer A	TSFA
00001	Increment A	INCA
00010	ADD A+B	ADD
00010	Subtract A-B	SUB
00110	Decrement A	DEC A
01000	AND A and B	AND
And so on……..		

Table below shows the microperation with different control words.

Table 1: Microoperation of CPU with control word

Micro operation	SELA	SELB	SELD	OPR	Control Word
	Symbolic Designation				
R1 ← R2 − R3	R2	R3	R1	SUB	010 011 001 00101
R4 ← R4 ∨ R5	R4	R5	R4	OR	100 101 100 01010
R6 ← R6 + 1	R6	-	R6	INCA	110 000 110 00001
R7 ← R1	R1	-	R7	TSFA	001 000 111 00000
Output ← R2	R2	-	None	TSFA	010 000 000 00000
Output ← Input	Input	-	None	TSFA	000 000 000 00000
R4 ← shl R4	R4	-	R4	SHLA	100 000 100 11000
R5 ← 0	R5	R5	R5	XOR	101 101 101 01100

3. Define stack. Explain the stack organization.

- Stack is a storage structure that stores information in such a way that the last item stored is the first item retrieved. It is based on the principle of LIFO (Last-in-first-out). The stack in digital computers is a group of memory locations with a register that holds the address of top of element. This register that holds the address of top of element of the stack is called ***Stack Pointer***.

Stack Operations

The two operations of a stack are:

1.**Push:** Inserts an item on top of stack.

2.**Pop:** Deletes an item from top of stack.

Implementation of Stack

In digital computers, stack can be implemented in two ways:

1.Register Stack

2.Memory Stack

Register Stack

- A stack can be organized as a collection of finite number of registers that are used to store temporary information during the execution of a program. The stack pointer (SP) is a register that holds the address of top of element of the stack.

Memory Stack

- A stack can be implemented in a random access memory (RAM) attached to a CPU. The implementation of a stack in the CPU is done by assigning a portion of memory to a stack operation and using a processor register as a stack pointer. The starting memory location of the stack is specified by the processor register as *stack pointer*. Computer memory partitioned into three segments: program, data and stack.

3. Explain the different instruction formats with examples.

……………..

<u>Three Address Instruction</u>

- Computers with three-address instruction formats can use each address field to specify either a processor register or a memory operand. The program in assembly language that evaluates X = (A + B) ∗ (C + D) is shown below, together with comments that explain the register transfer operation of each instruction

ADD R1, A, B   R1 ← M [A] + M [B]

ADD R2, C, D $R2 \leftarrow M[C] + M[D]$
MUL X, R1, R2 $M[X] \leftarrow R1 * R2$

- It is assumed that the computer has two processor registers, R1 and R2. The symbol M [A] denotes the operand at memory address symbolized by A.
- The advantage of the three-address format is that it results in short programs when evaluating arithmetic expressions. The disadvantage is that the binary-coded instructions require too many bits to specify three addresses.

Two address instructions

- Two address instructions are the most common in commercial computers. Here again each address field can specify either a processor register or a memory word. The program to evaluate $X = (A + B) * (C + D)$ is as follows:

MOV R1, A $R1 \leftarrow M[A]$
ADD R1, B $R1 \leftarrow R1 + M[B]$
MOV R2, C $R2 \leftarrow M[C]$
ADD R2, D $R2 \leftarrow R2 + M[D]$
MUL R1, R2 $R1 \leftarrow R1*R2$
MOV X, R1 $M[X] \leftarrow R1$

- The MOV instruction moves or transfers the operands to and from memory and processor registers. The first symbol listed in an instruction is assumed to be both a source and the destination where the result of the operation is transferred.

One Instruction Format

- One-address instructions use an implied accumulator (AC) register for all data manipulation. For multiplication and division there is a need for a second register. However, here we will neglect the second and assume that the AC contains the result of tall operations. The program to evaluate $X = (A + B) * (C + D)$ is

LOAD A $AC \leftarrow M[A]$
ADD B $AC \leftarrow A[C] + M[B]$
STORE T $M[T] \leftarrow AC$
LOAD C $AC \leftarrow M[C]$
ADD D $AC \leftarrow AC + M[D]$
MUL T $AC \leftarrow AC * M[T]$
STORE X $M[X] \leftarrow AC$

- All operations are done between the AC register and a memory operand. T is the address of a temporary memory location required for storing the intermediate result.

Zero Address Instruction

- A stack-organized computer does not use an address field for the instructions ADD and MUL. The PUSH and POP instructions, however, need an address field to specify the operand that communicates with the stack. The following program shows how $X = (A + B) * (C + D)$ will be written for a stack organized computer. (TOS stands for top of stack)

PUSH A $TOS \leftarrow A$
PUSH B $TOS \leftarrow B$
ADD $TOS \leftarrow (A + B)$
PUSH C $TOS \leftarrow C$
PUSH D $TOS \leftarrow D$
ADD $TOS \leftarrow (C + D)$
MUL $TOS \leftarrow (C + D) * (A + B)$
POP X $M[X] \leftarrow TOS$

- To evaluate arithmetic expressions in a stack computer, it is necessary to convert the expression into reverse Polish notation. The name “zero-address” is given to this type of computer because of the absence of an address field in the computational instructions

4. Explain the different types of instruction addressing modes.

➢ The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced.

1. **Implied Mode :** In this mode the operands are specified implicitly in the definition of the instruction. For example:- CMA - “complement accumulator” is an implied-mode instruction because the operand in the accumulator register is implied in the definition of the instruction. In fact, all register reference instructions that use an accumulator are implied-mode instructions.

Advantage: no memory reference. Disadvantage: limited operand

2. **Immediate Mode :** In this mode the operand is specified in the instruction itself. In other words, an immediate-mode instruction has an operand field rather than an address field. · This instruction has an operand field rather than an address field. The operand field contains the actual operand to be used in conjunction with the operation specified in the instruction. · These instructions are useful for initializing register to a constant value; For example MVI B, 50H

Advantage: no memory reference. Disadvantage: limited operand

3.Register direct addressing mode: In this mode, the operands are in registers that reside within the CPU. The particular register is selected from the register field in the instruction. For example MOV A, B

Advantage: no memory reference. Disadvantage: limited address space

4. **Register Indirect Mode**: In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in the memory. In other words, the selected register contains the address of the operand rather than the operand itself. Before using a register indirect mode instruction, the programmer must ensure that the memory address of the operand is placed in the processor register with a previous instruction. For example LDAX B

Advantage: Large address space. The address field of the instruction uses fewer bits to select a register than would have been required to specify a memory address directly. Disadvantage: Extra memory reference

5. Auto increment or Auto decrement Addressing Mode: This is similar to register indirect mode except that the register is incremented or decremented after (or before) its value is used to access memory. When the address stored in the registers refers to a table of data in memory, it is necessary to increment or decrement the registers after every access to the table. This can be achieved by using the increment or decrement instruction. In some computers it is automatically accessed. The address field of an instruction is used by the control unit in the CPU to obtain the operands from memory. Sometimes the value given in the address field is the address of the operand, but sometimes it is the address from which the address has to be calculated.

6. Direct Addressing Mode : In this mode the effective address is equal to the address part of the instruction. The operand resides in memory and its address is given directly by the address field of the instruction. For example LDA 4000H
Advantage: Simple. Disadvantage: limited address field
7. **Indirect Addressing Mode:** In this mode the address field of the instruction gives the address where the effective address is stored in memory. Control unit fetches the instruction from the memory and uses its address part to access memory again to read the effective address.
Advantage: Flexibility. Disadvantage: Complexity
8. Displacement Addressing Mode : A very powerful mode of addressing combines the capabilities of direct addressing and register indirect addressing. The address field of instruction is added to the content of specific register in the CPU.
. Effective Address (EA) = R + Add of instruction
Advantage: Flexibility. Disadvantage: Complexity
9. Relative Addressing Mode: In this mode the content of the program counter (PC) is added to the address part of the instruction in order to obtain the effective address. The address part of the instruction is usually a signed number. When the number is added to the content of the program counter, the result produces an effective address whose position in memory is relative to the address of the next instruction. Effective Address (EA) = PC + A
9. Indexed Addressing Mode · In this mode the content of an index register (XR) is added to the address part of the instruction to obtain the effective address. · The index register is a special CPU register that contains an index value. In computing, a **base address** is an address serving as a reference point ("base") for other addresses. · Note: If an index-type instruction does not include an address field in its format, the instruction is automatically converted to the register indirect mode of operation. Effective Address (EA) = XR + A
10.Base Register Addressing Mode · In this mode the content of a base register (BR) is added to the address part of the instruction to obtain the effective address. · This is similar to the indexed addressing mode except that the register is now called a base register instead of the index register. · The base register addressing mode is used in computers to facilitate the relocation of programs in memory i.e. when programs and data are moved from one segment of memory to another. Effective Address (EA) = BR + A

5. Explain the different Data Transfer and Manipulation instruction.

- Most computer instructions can be classified into three categories:
  1)Data transfer,
  2) Data manipulation,
  3) Program control instructions

Data Transfer Instruction

Data transfer instructions move data from one place in the computer to another without changing the data content. The most common transfers are between memory and processor registers, between processor registers and input or output, and between the processor registers themselves.
Typical Data Transfer Instruction :
Load : transfer from memory to a processor register, usually an AC (*memory read)*
Store : transfer from a processor register into memory (*memory write)*
Move : transfer from one register to another register
Exchange : swap information between two registers or a register and a memory word

Input/Output : transfer data among processor registers and input/output device
Push/Pop : transfer data between processor registers and a memory stack

Data Manipulation Instructions perform operations on data and provide the computational capabilities for the computer.
It is divided into three basic types:

I. Arithmetic,
II. Logical and bit manipulation,
III. Shift Instruction

Arithmetic,

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

Logical and Bit Manipulation

NAME	Mnemonic
Clear	CLR
Complement	COM
AND	AND
OR	OR
Exclusive-or	XOR
Clear carry	CLRC
Set carry	SETC
complement carry	COMC
Enable interrupt	EI
Disable interrupt	DI

Shift Instuction

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

6. **What are control instructions? Explain the different types of program control instructions with their roles.**

- Program control instructions specify conditions for altering the content of the program counter, while data transfer and manipulation instructions specify conditions for data-processing operations.
- The most basic kind of program control is the **unconditional branch** or **unconditional jump**. **Branch** is usually an indication of a short change relative to the current program counter. **Jump** is usually an indication of a change in program counter that is not directly related to the current program counter (such as a jump to an absolute memory location or a jump using a dynamic or static table), and is often free of distance limits from the current program counter.
- The pentultimate kind of program control is the **conditional branch** or **conditional jump**. This gives computers their ability to make decisions and implement both loops and algorithms beyond simple formulas.
- Most computers have some kind of instructions for **subroutine call** and **return** from subroutine

NAME	Mnemonic
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RET
Compare(by subtraction)	CMP
Test(by ANDing)	TST

7. **Define program interrupt. Explain the types of interrupt.**

- Program interrupt refers to the transfer of program control from the currently running program to another service program as a result of an external or internal generated request. Control returns to the original program after the service program is executed.
- The different typesof interrupt are :

i. External Interrupt
ii. Internal interrupt
iii. Software Interrupt

External Interrupt

- External Interrupt come from the input output devices, from a timing device , from a circuit monitoring the power supply , or from any other external sources .
- Examples : I/0 devices requesting transfer of data , I/O devices finished transferring data , power failure ,elapsed time of event

Internal interrupt

- Internal Interrupt arises from illegal or erroneous use of the instruction or data. Internal interrupts are also called traps. Example of if internal interrupt are register overflow, attempt to divide by zero, an invalid code , stack overflow , and protection violation. The service program that processes the internal interrupt determines the corrective measures to be taken.

Software Interrupt

- A software interrupt is initiated by executing the instruction. (INT or RST instruction). Software interrupt is a special call instruction that behaves like an interrupt rather than a subroutine call. It can be used by the programmer to initiate an interrupt rather than a subroutine call.

8. Write short notes on: Status bit conditions, Conditional branch instructions, Subroutine Call and Return.

Status Bit Condition :

- Most CPU architectures maintain a number of status bits that indicate the results from the most recent ALU operation. It is convinent to supplement the ALU circuit in the CPU with a status register where status bit condition can be stored for further analysis. Status bits are also called condition code bit or flag bit.These bits are usually stored in a *status register*, which is not directly accessible as an argument in machine instructions. bits are set or cleared as a result of an operation performed in the ALU

The 4 status bits are symbolized by C, S, Z, and V, and are defined as:

i. Bit C (carry-out) is set to 1 only when the end-carry of the ALU operation is 1. It is cleared to 0otherwise.
ii. Bit S (sign) is set to the value of the left-most bit of the ALU output.
iii. Bit Z (zero) is set to 1 only when the ALU output data bits are all 0's. It is cleared to 0 otherwise.
iv. Bit V (overflow) is set to 1 only when the last two carries (C7 and C8) of the arithmetic units are different.

The status bits are used to evaluate the branching conditions of conditional program control instructions.

```
                      1111111
A   11111111    A      11111111
B   01111111    B'+1    10000001
--------------------------------
A-B             1 10000000
```

V = 0 (c7 xor c8: clever way of predicting wrong sign on result)
Z = 0 (o7'o6'...o0')
S = 1 (o7)
C = 1 (c8)

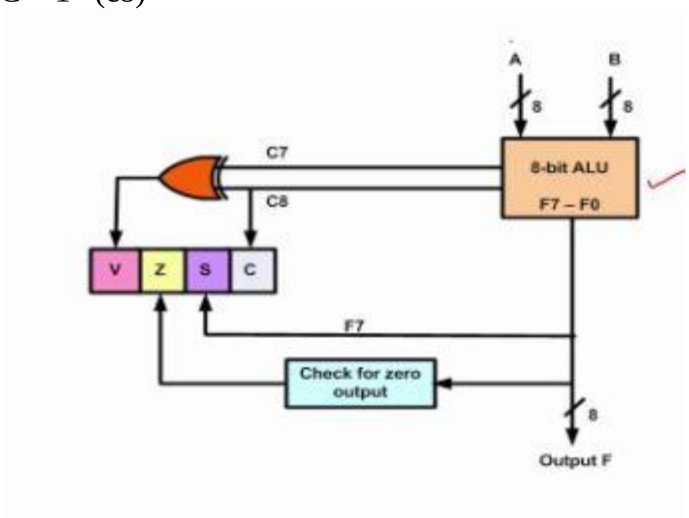


Figure 1: An 8 bit ALU wiht 4 bit register

Conditional Branch Instruction

- A branch is an instruction in a computer program that can cause a computer to begin executing a different instruction sequence and thus deviate from its default behavior of executing instructions in order. A branch instruction can be either an unconditional branch or a conditional branch. Conditional branch which may or may not cause branching, depending on some condition. Branch instructions are used to implement control flow in program loops and conditionals (i.e., executing a particular sequence of instructions only if certain conditions are satisfied).The conditional branch instruction checks the status of status bit (like C,Z,S,O) for checking branching occurs or not. Some of the Conditional branching instruction are as follows:

Mnemonic	Branch condition	Tested condition
BZ	Branch if zero	$Z = 1$
BNZ	Branch if not zero	$Z = 0$
BC	Branch if carry	$C = 1$
BNC	Branch if no carry	$C = 0$
BP	Branch if plus	$S = 0$
BM	Branch if minus	$S = 1$
BV	Branch if overflow	$V = 1$
BNV	Branch if no overflow	$V = 0$
Unsigned compare conditions $(A - B)$		
BHI	Branch if higher	$A > B$
BHE	Branch if higher or equal	$A \geq B$
BLO	Branch if lower	$A < B$
BLOE	Branch if lower or equal	$A \leq B$
BE	Branch if equal	$A = B$
BNE	Branch if not equal	$A \neq B$
Signed compare conditions $(A - B)$		
BGT	Branch if greater than	$A > B$
BGE	Branch if greater or equal	$A \geq B$
BLT	Branch if less than	$A < B$
BLE	Branch if less or equal	$A \leq B$
BE	Branch if equal	$A = B$
BNE	Branch if not equal	$A \neq B$

Subroutine Call and Return

- Subroutine is self-contained sequence of instructions that performs a given computational task. During the execution of a program, a subroutine may call many times at various points in the main program. Each time the subroutine is called, a branch is executed at the beginning of the subroutine to start executing its sets of instruction. After the subroutine is executed, a branch is made back to the main program. A call subroutine instruction consists of an operational code with an address that specifies the beginning of the subroutine.

  The instruction is executed by performing following operation.

- The address of the next instruction available in the program counter is stored in a temporary location so the subroutine knows where to return.
- Control is transferred to the beginning of the subroutine. Then return from subroutine transfers the return address from the temporary location into the program counter.
- A subroutine call can be implemented with the following microopearation:

  SP← SP-1 Decrement stack pointer

M[SP] ← PC   Push content of PC onto the stack
PC← effective Address of subroutine

- The instruction that returns from the last subroutine is implemented using following microoperation
  PC ←M[SP]   Pop stack and transfer to pc
  SP← SP+1   Increment stack pointer

10. Differentiate between CISC and RISC architecture.

RISC→ **Reduced instruction set computers**

CISC→**Complex instruction set computers**

1. In RISC the instruction set size is small while in CISC the instruction set size is large.
2. RISC uses fixed format (32 bits) and mostly register-based instructions whereas CISC uses variable format ranges from 16-64 bits per instruction.
3. RISC uses a single clock and limited addressing mode (i.e., 3-5). On the other hand, CISC uses multi-clock 12 to 24 addressing modes.
4. The number of general purpose registers that RISC uses ranges from 32-192. On the contrary, CISC architecture uses 8-24 GPR's.
5. Register-to-register memory mechanism is used in RISC with independent LOAD and STORE instructions. In contrast, CISC uses memory to memory mechanism for performing operations, furthermore, incorporated LOAD and STORE instructions.
6. RISC has split data and instruction cache design. As against, CISC uses unified cache for data and instructions, although latest designs also use split caches.
7. Most of the CPU control in RISC is hardwired without having a control memory. Conversely, CISC is microcoded and uses control memory (ROM), but modern CISC also uses hardwired control.