COMPUTER ORGANIZATION&ARCHITECHTURE

Module -2

Central processing unit

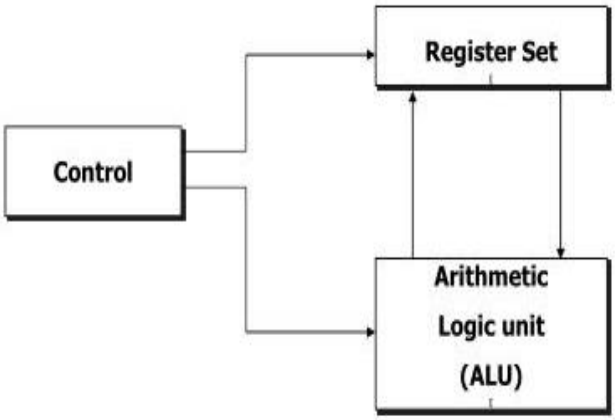
The part of the computer that performs the bulk of data-processing operations is called the central processing unit and is referred to as the CPU. The CPU is made up of three major units

- Register set
- Arithmetic logic unit (ALU)
- Control unit

➢Register set – The register set stores intermediate data used during used during the execution of the instructions

➢ALU – performs the required micro operations for executing the instructions.

➢Control unit – Supervises the transfer of information among the registers and instructs the ALU as to which operation to perform.



Major components of CPU

General Register Organization

A bus organization for 7 CPU registers. The output of each register is connected to two multiplexers (MUX) to form the two buses A and B. The selection lines in each multiplexer select one register or the input data for the particular bus. The A and B buses from the input to a common Arithmetic Logic Unit (ALU). The operation selected in the ALU determines the Arithmetic Logic micro operation to be performed. The result of the micro operation is available for output data and also goes into the inputs of all the registers. The register that receives the information from the output bus is selected by a decoder.

The control unit that operates the CPU bus system directs the information flow through the registers and ALU by selecting the various components in the system.

example: $R_1 \leftarrow R_2 + R_3$

1. MUX A selector (SELA): to place the content of R_2 into bus A

2. MUX B selector (SELB): to place the content of R_3 into bus B

3. ALU operation selector (OPR): to provide the Arithmetic addition A+B

4. Decoder destination selector(SELD): to transfer the content of the output bus into R_1.

This four control selection variables are generated in the control unit and must be available at the beginning of a clock cycle.

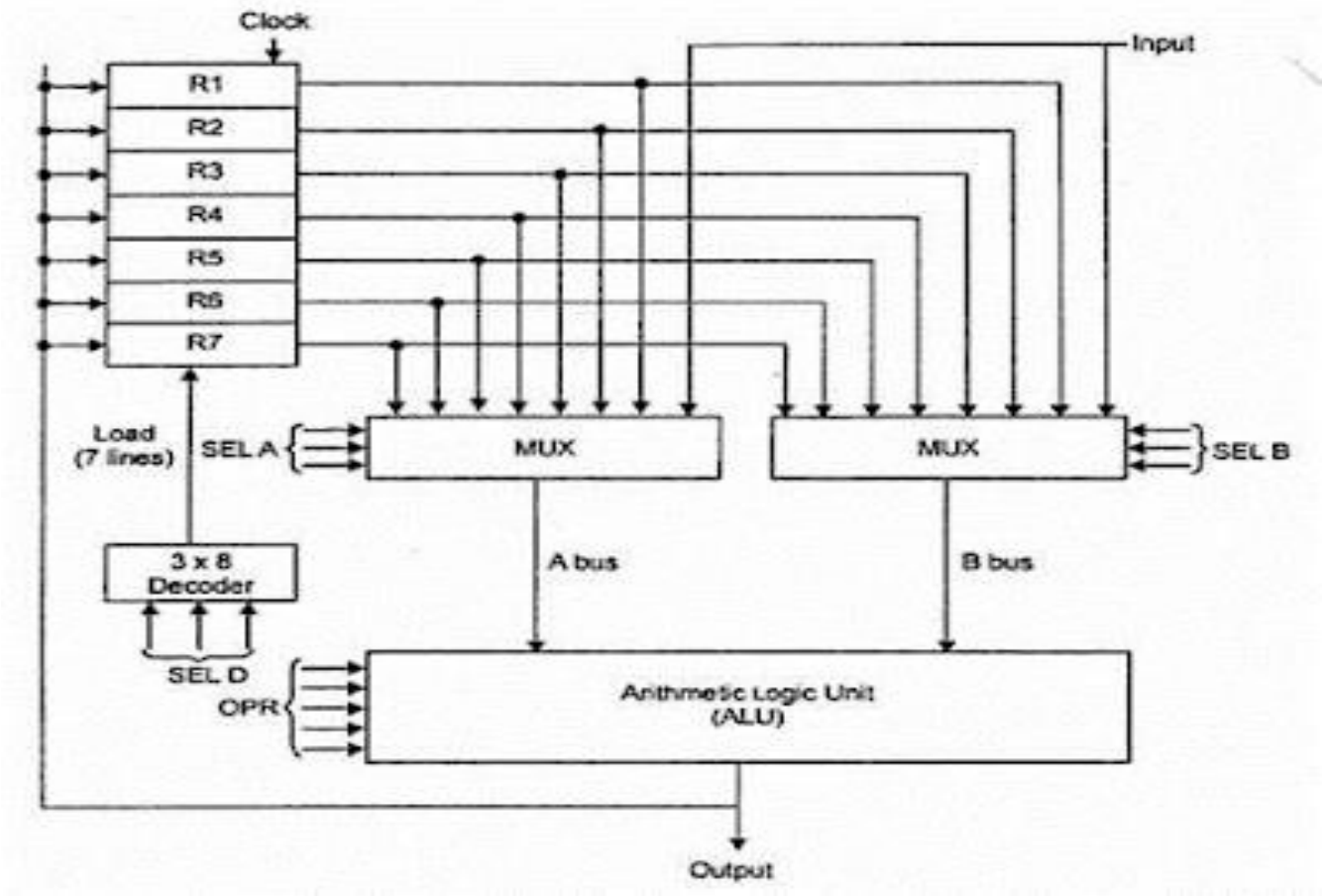
Control word

There are 14 binary selection inputs in the unit, and their combined value specifies a control word.

Three fields contains 3-bits each and one field have five bits.

3	3	3	5
SELA	**SELB**	**SELD**	**OPR**

Control word



General Register Organization

Stack Organization

A useful feature that is included in the CPU of the most computers is stack or last in first out (LIFO) list. A stack is a storage device that stores information in such a manner that the item stored. Last is the first item retrieved.

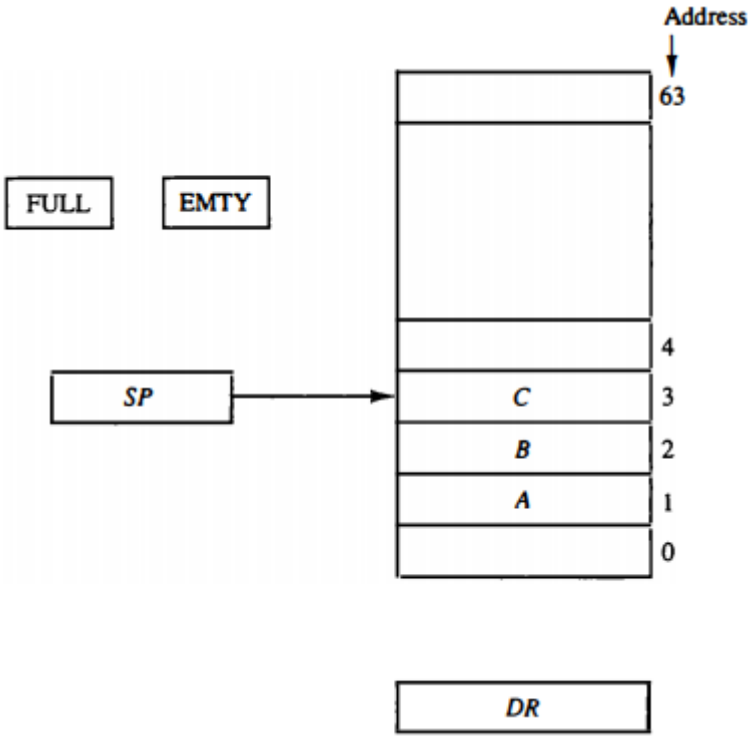
The register that holds the address for the Stack is called Stack pointer (SP), because its value always points at the top item in the stack.

The two operations of the stack are the insertion and the deletion of items. The operation of insertion is called push. The operation of deletion is called pop.

Register Stack

A stack can be placed in the portion of a large memory or it can be organized as the collection of a finite no. of words or registers. The stack pointer register SP contains a binary number whose value is equal to the address of the word that is currently on top of stack.

In a 64-bit word stack, the stack pointer contains six bits because $2^6 = 64$ since SP has only six bits, it cannot exceed a number greater than 63.



Block diagram of 64-word stack

Initially, sp is cleared to 0, EMTY is set to 1, and FULL is cleared to 0, so that Sp points to the word at address 0 and the stack is marked empty and not full.

Memory Stack

The implementation of 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 memory partitioned into three segments.

- Program
- Data
- Stack

The program counter pc points at the address of the next instruction in the program. The address register AR points at an array of data. The stack pointer points at the top of the stack. The three registers are connected to common address bus, and either one can provide an address for the memory. PC is used during the fetch phase to read an instruction .AR is used during the execute phase to read an operand. SP is used to push or pop items into or from the stack.

Reverse polish Notation

A stack organization is very effective for evaluating arithmetic expressions. There are mainly three types of representations.

1. A+B Infix Notation
2. +AB prefix or polish Notation
3. AB+ past fix or Reverse polish Notation

Eg : A*B+C*D

Reverse polish Notation: AB*CD*+

➢Refer example of Stack operation 3*4+5*6

Addressing modes

The operation field of an instruction specifies the operation to be performed. The operation must be executed on some data stored in computer registers or memory words. The way the operands are chosen during program execution is dependent on the addressing modeof the instruction. The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced.

Computer use addressing mode techniques for the purpose of accommodating one or both of the following.

1. To give the programming versatility to the user

2. To reduce the number of bits in the addressing field of the instruction.

The operation code specifies the operation to be performed. The mode field is used to locate the operands needed for the operation. If there is an address field in the instruction, it may designate a memory address or processor register.

➢Although most addressing modes modify the address field of the instruction, there are two modes that needs no address field at all.

opcode	**Mode**	**Address**

Implied Mode

In this mode the operands are specified implicity in the definition of the instruction. Eg : complement accumulator is an implied mode instruction. Because the operand in the accumulator register is implied in the definition of instruction.

Immediate mode

In this mode the operand is specified in the instruction itself. An immediate mode instruction has an operand field rather than address field. The operand field contains the actual operand to be used in the conjunction with the operation specified in the instruction.

Register mode

In this mode the operands are in registers that reside within the CPU. The particular register is selected from a register field in the instruction. Register contains the address of the operand rather that operand itself.

Register indirect mode

In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in memory. The selected register contains the address of the operand rather than the operand itself.

Auto decrement or Auto increment Mode

This is similar to the register indirect mode except that the register is incremented or decremented after its value is used to access memory. When the address stored in the register refers to a table of data in memory, it is necessary to increment or decrement the register.

- To differentiate among the various addressing modes it is necessary to distinguish between address part of the instruction and the effective address used by the control when executing the instruction. The effective address is defined to be the memory address obtained from the computation dictated by the given addressing mode. The effective address is the address of the operand in the computational type instruction.

Direct Address mode

In this mode the effective address is equal to the address part of the instruction. The operand resides in the memory and its address is given directly by the address field of the instruction.

Indirect Address Mode

In this mode the address field of the instruction gives the address where the effective address is stored in memory.

Effective address = Address part of instruction + Content of CPU register

Relative Address Mode

In this mode the content of the program counter is added to the address part of the instruction in order to obtain the effective address.

Indexed Addressing Mode

In this mode the content of an index register 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.

Base Register Addressing Mode

In this mode the content of a base register is added to the address part of the instruction to obtain the effective address.

Instruction Classification

1. Data Transfer Instruction
2. Data Manipulation Instruction
3. Program Control Instruction

Data Transfer Instruction

(Data Transfer Instruction cause transfer of data from one location to another without changing the binary information content). The most common transfers are between memory and processor registers, between processor registers, input or output registers and between the processor registers itself.

Typical Data Transfer Instruction

1. Load LD	4. Exchange XCH	7. Push PUSH
2. Store ST	5. Input IN	8. Pop POP
3. Move MOV	6. Output OUT	

The load instruction has been used mostly to designate a transfer from memory to processor register, usually an accumulator. The store instruction designate a transfer from processor register to memory. The move instruction has been used in computers with multiple CPU registers to designate a transfer between one register to another. The exchange instruction swaps information between two registers or a register and a memory word. The input and output instructions transfer data among processor registers and Input or Output terminals. The push and pop instructions transfer data between processor register and a memory stack.

Data Manipulation Instructions

Performs Operations on Data and provide the computational capabilities for the computer. It is divided into 3 basic types

- Arithmetic Instructions
- Logical and Bit manipulation Instructions
- Shift Instruction

Arithmetic Instructions

The four basic arithmetic operations are addition, subtraction, multiplication and division. Most computers provide instructions for all four operations. The instruction adds 1 to the value stored in a register or a memory word. The decrement instruction subtracts 1from the value stored in a register or a memory word.

Typical Arithmetic Instruction

- Increment INC
- Decrement DEC
- Add ADD
- Subtract SUB
- Multiply MUL
- Divide DIV
- Add with Carry ADDC
- Subtract with borrow SUBB
- Negate(2's Compliment) NEG

Logical And Bit manipulation Instructions

Logical Instructions perform binary operations on strings of bits stored in registers. They are useful for manipulating the individual bits or a group of bits that represent binary coded information. The AND, OR and XOR instructions produce the corresponding logical operations on individual bits of operands. There are 3 bit manipulation instructions possible: a selected bit can be cleared to 0, or can be set to 1, or can be complemented. The AND instruction is used to clear a bit or a selected bits of an operand.

Typical Logical and Bit Manipulation Instruction

- Clear CLR
- Compliment COM
- AND AND
- OR OR
- Exclusive-or XOR
- Clear Carry CLRC
- Set Carry SETC
- Complement Carry COMC
- Enable Interrupt EI
- Disable Interrupt DI

Shift Instructions

Instructions to shift the content of an operand are quite useful and are often provided in several variations. Shifts are operations in which the bits of a word are moved to left or right. The bit shifted in at the end of the word determines the type of shift used. Shift instructions may specify either logical shifts, arithmetic shifts or rotate type operations. In either case the shift may be right or to left.

Typical Shift Instructions

- 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

Program Control

Instructions are always stored in successive memory locations. When processed in the CPU, the instructions are fetched from consecutive memory locations and executed. Each time an instruction is fetched from memory the program counter is incremented so that it contains the address of the next instruction in sequence. The branch and jump instructions are used interchangeably to mean the same thing but sometimes they are used to denote different addressing modes.

Typical Program Control Instructions

- Branch BR
- Jump JMP
- Skip SKP
- Call CALL
- Return RET
- Compare(by subtraction) CMP
- Test (by adding) TST

The skip instruction does not need an address field and is therefore a zero-address instruction. A conditional skip instruction will skip the next instruction if the condition is met.

Status Bit Condition

It is sometimes convenient to supplement the ALU circuit in the CPU with a status register where status conditions can be stored for further analysis. Status bits are also called condition code bits or flag bits. The four Status bits are symbolized by C, S, Z and V. The bits are set or cleared as a result of an operation to be performed in the ALU.

1. Bit C (carry) is set to 1 if the end carry C_8 is 1. It is cleared to 0 if the carry is 0.

2. Bit S (sign) is set to 1 if the highest order bit F_7 is 1. It is set to 0 if the bit is 0.

3. Bit Z (zero) is set to 1 if the output of ALU contains all 0's. It is cleared to 0 otherwise.

4. Bit V (overflow) is set to 1 if the exclusive OR of the last two carries is equal to 1.

Conditional Branch Instruction

Each mnemonic is constructed with letter B (for branch) and an abbreviation of the condition name. When the opposite condition state is used, the letter N (for No) is inserted to define the 0 state.

Subroutine Call and Return

A subroutine is a self-contained sequence of instructions that performs a given computational task. During the execution of a program a subroutine may be called to perform its function many times at various points in the main program.

The instruction that transfers program control to a subroutine is known by different names. The most common names used are call subroutine, jump to subroutine, branch to subroutine or branch and save address. The instruction is executed by performing two operations.

1. The address of the next instruction available in the program counter is stored in a temporary location

2. Control is transferred to the beginning of subroutine.

Program Interrupt

The concept of program interrupt is used to handle a variety of problems that arise out of normal program sequence. Program interrupt refers to transfer of program control from a currently running program to another service program as a result of an external or internal

generated request, control returns to the original program after the service program is executed.

1. The interrupt is usually initiated by an internal or external signal rather than from the execution of an instruction.

2. The address of the interrupt service program is determined by the between rather than from the address field of an instruction.

The state of the CPU at the end of the execute cycle is determined from

1. The content of the program counter

2. The content of all processor registers

3. The content of certain status conditions

Type of Interrupts

1. External interrupts: Come from Input-output devices from a timing device, from a circuit monitoring the power supply or from other external source.

2. Internal Interrupts: arise from illegal or erroneous use of an instruction or data. Internal interrupts are also called traps

3. Software Interrupt: A software interrupt is initiated by executing an instruction. S/W interrupt is a special call instruction that behaves like an interrupt rather than a subroutine call