K.T.S.P.Mandal's

Hutatma Rajguru Mahavidyalaya, Rajgurunagar

Tal-Khed, Dist.-Pune 410505.

F.Y.BSc (Computer Science)

Semester-II

Electronics Paper-II

Subject- ELC-122 – Basics of Computer Organization

According to new CBCS syllabus w.e.f.2019-2020

Prof.Aparna P.Kulkarni

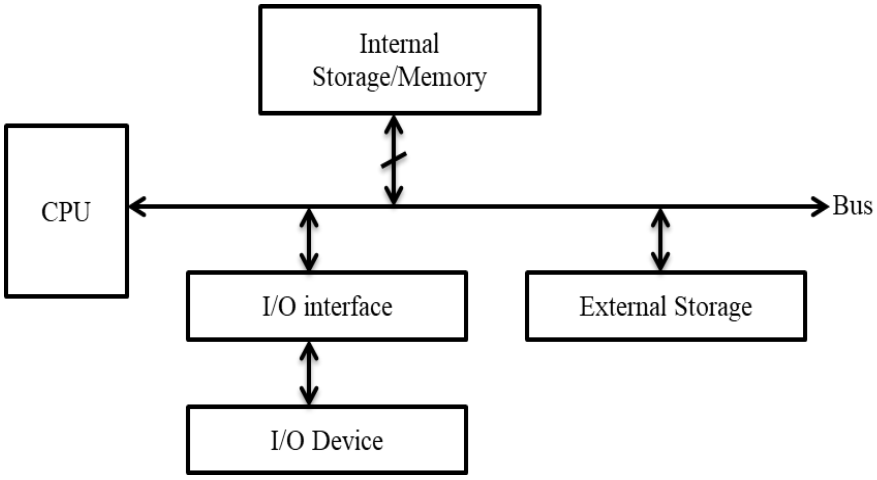
Department of Computer Science

Hutatma Rajguru Mahavidyalaya, Rajgurunagar.

Unit-III- Basics of Computer System

- Basics Computer Organization, Concept of Address bus, Data Bus, Control Bus,
- CPU Block Diagram and Explanation of each block
- Register based CPU Organization
- Concept of Stack & its Organization
- I/O Organization: Need of interface, block diagram of general I/O interface.

Basics Computer Organization:



- A CPU is the heart of a computer system. Its job involves interacting with memory and I/O devices for transfer of information. It is also responsible for coordinating and synchronizing the operators of various hardware blocks of the computer system and also for carrying out various arithmetic and logical operations on the data.
- The internal storage is provided by the RAM and ROM. All the operating system programs and other very important system information, lookup tables etc. are stored in the ROM.
- The RAM will contain the user program which needs to be processed. The RAM also is a part of the main memory of a system.
- The various I/O devices are connected to the CPU through interfaces. The interfaces help as a bridge to connect the CPU with I/O devices. The interfaces provide storage space to temporarily store information, provide various signals to communicate with CPU and I/O devices. They are also responsible for proper data conversions and conversion of data formats as understood by the CPU.
- All these hardware blocks are inter-connected to one another through buses. Information such as address, data and control information flow on these buses.

Concept of Buses:

A bus is a set of conducting lines/paths which interconnect different parts of a computer system and are responsible for carrying out different types of information.

Depending of the type of information being carried bus can be three types:

i. Address Bus

ii. Data Bus

iii. Control Bus

i. Address Bus:

- An address bus is used to carry addresses generated by the CPU. These addresses either select memory locations or various I/O devices as per need. The direction of this bus is unidirectional always coming out of CPU.
- The size of the address bus decides the addressing capability of the CPU. If 'n' is the width in bits of the address, bus the addressing capacity is 2^n.
- This means a CPU with 8 bit address bus can generate only 2^8 unique addresses that are 256. Assuming each address selects, memory location, a memory of maximum 256 bytes can be addressed.
- If the numbers of address lines are 32, then 2^{32} unique addresses are generated. So one can connect memories of up to 4GB capacity to the computer system. From the available addresses, some of the addresses can be reserved to select I/O devices.

ii. Data Bus:

A Data bus is used to carry actual information between CPU and to devices or memory.

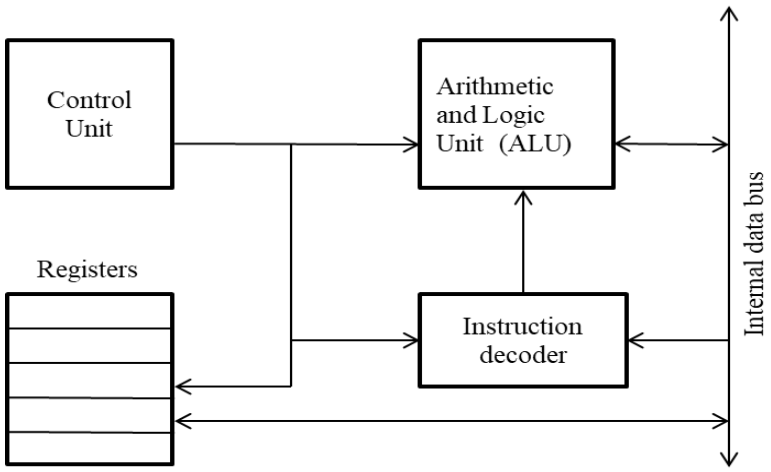
- Since information can flow in both directions, i.e., CPU to output device and input device to CPU, the data bus is bidirectional.
- Increasing the size of the data bus increases the data transfer rate of the CPU. So more information can be exchanged by the CPU per unit time.

iii. Control Bus:

- The control bus carries control information that commands and controls the operation of the I/O interfaces and other hardware units in a computer system.
- It also carries the various status information in the form of acknowledgements from various devices. So the control bus will also be bidirectional.

However when information flows through the buses, all of these information, i.e., address, data and control are not simultaneously generated. Each information is generated at a specific instant of time and independently. Also all these information is not always available.

Block Diagram of CPU:



- CPU is a very important block of a computer system. Right from address generation, generating control signals, sequencing of control signals, providing appropriate timings and carrying out various logical and arithmetic operations are jobs of the CPU. In order to do these, there must be various specialized hardware blocks inside the CPU.
- The control unit is responsible for coordinating and synchronizing operations of various hardware blocks in the CPU and the blocks connected to the CPU. For this purpose it needs to generate different control signals.
- The control signals have to be generated in specific sequences and every instruction has a different sequence of control signals. So sequencing of control signals have to be carried out by the CPU. Each signal has to be active for specific time duration. So providing timings to the signals and providing time to the various blocks to carry out the jobs assigned to them is also a very important job of the CPU.
- The ALU or the arithmetic and logic unit is responsible to carry out different arithmetic and logical operations on the operands sent to the ALU. It provides the output with the desired precision especially during floating point operations.

- An instruction decoder is responsible for decoding the instructions which are fetched from the memory. Programs are stored in memory. When it is to be executed, instructions are fetched one by one from memory into instruction decoder. Understanding where the operands are and what operation is to be performed on them is done by the instruction decoder. Then the control unit takes over to generate various control signals required to execute or complete the instruction.
- Inside the CPU there are a few registers which act as temporary storage for operands before they are processed by the ALU. They also hold the results from the ALU temporarily. There registers are high speed memories to transfer information fast.

Internal bus connects all these blocks inside the CPU.

Registers in CPU:

- Every CPU has a set of registers to hold information. These registers can be classified as program visible and program invisible.
- A program visible register can be accessed by the user through appropriate software instruction. A program invisible instruction is not accessible to user in any case. These are not instructions available to access them. However the CPU has free access to these registers.
- The program visible registers can be further classified as General Purpose Registers (GPR) and special function registers. The GPR registers can be used to store any type of information may it be address or data. These registers can support storage of 32 bit information as incase of extended register 16 bit information as word registers or 8 bit information like a byte register.
- A special function register is used by the CPU to store information which help the way CPU should function. The information in these registers control the operation of various hardware blocks inside the CPU. Some of the common SFR are Program Counter Register it is a pointer register. It points to or holds the address of the instruction that is to be executed next.
    i. Stack Pointer Register: The contents of this register point to the top of the stack. It tells the CPU where the next information should be pushed or popped out from stack.
    ii. Instruction Register: It holds the instruction that has been fetched from memory. This is because the CPU could be executing an instruction when the new instruction is fetched from memory.

iii. Instruction Pointer Register: It points to the instruction that is to be brought for execution.

iv. Memory Address Register (MAR): The contents of this register is an address and it points to the memory location from where the data/operand/information is to be fetched.

v. Memory Buffer Register (MBR): This register holds the data that has to be fetched from the selected memory location or that has to be stored in the selected location.

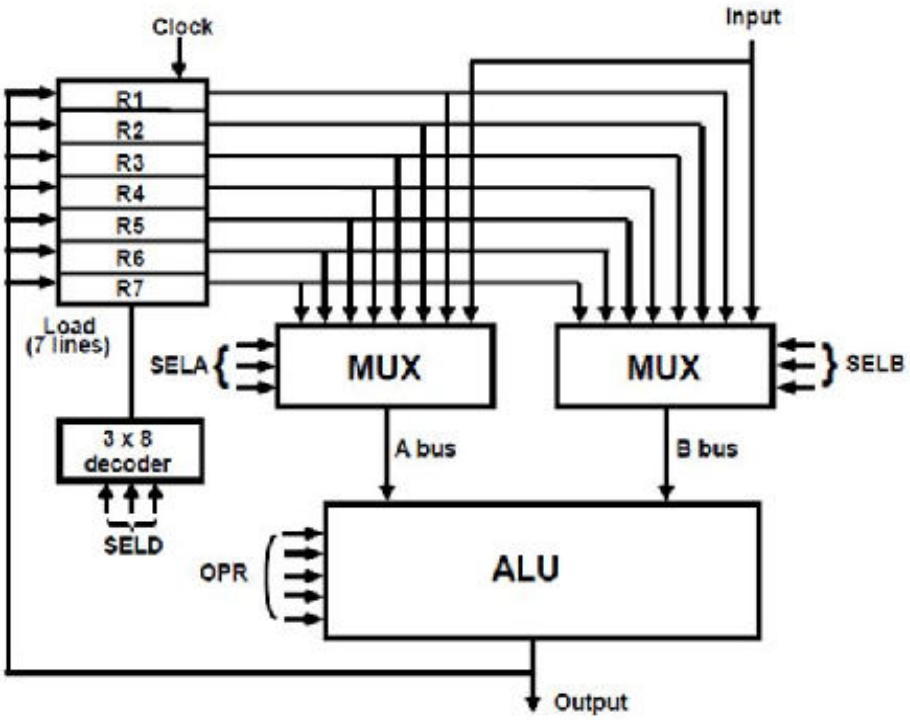
vi. Data Register (DR): This register holds data that is to be pushed onto stack or that has been popped out of stack.

vii. Flags: This register is a collection of 1 bit flip-flops. Each flipflop indicates certain condition that come into existence upon execution of an instruction. Examples of various conditions of the result are sign (+ve or -ve), zero (result is 0 or non zero), carry (result generated a carry or not) parity (result has odd or even parity) etc.

Register Based CPU Organization:

- Usually memory stores data, return addresses during a subroutine call, pointer and counters, operands and results. However access to memory is very time consuming because memory accesses are slow. It leads to lots of wastage of time thus affecting the overall execution time of instructions and hence the program itself.
- This waiting time for fetching operands can be reduced if the information is stored in temporary high speed units called the registers. Access to these registers is very fast.
- Every CPU is therefore having a set of General Purpose Registers (GPR) for this purpose.
- These registers are connected through a common bus. These registers communicate with one another for data transfer as well as are involved during performing of various micro operations.
- There should be a provision for selecting source registers for reading operands stored as well as destination register to store results.

The diagram given below shows how these registers are connected inside the CPU, Consider a CPU with 7 GPR (General Purpose Registers) and an external source like memory for holding the operands.



- Assuming that for every operation performed by ALU, 2 source operands will be required. One through ‘A’ bus and other 'B' bus. The operands for the A and B buses are selected from 8 different sources, seven GPR and one external input. This is possible using 2 multiplexers ‘A’ and 'B'. Multiplexer ‘A’ selects one of the 8 source inputs and puts it on ‘A’ bus. It will have 3 control signals which will decide the selection of inputs. Similarly multiplexer 'B' selects one of the 8 source inputs and sends it on 'B' bus.
- The ALU of the CPU can perform many arithmetic and logic operations on the operands send to it. Which operation should be performed by the ALU is decided by the operation select lines. If the ALU can perform 128 different operations then 7 instruction select lines are required $2^7 = 128$).
- After finishing an operation, the ALU produces the result. This result can again be stored back in these 7 GPR or external output or memory. For selecting the destination, a 3 to 8 address decoder is used. There will be 8 outputs of the decoder which will enable one of the 8 destinations and the result is guided into the selected destination.

The source and destination can be selected with the following control line combinations/addresses.

For source/destination	3 bit control/address		
R_1	0	0	1
R_2	0	1	0
R_3	0	1	1
R_4	1	0	0
R_5	1	0	1
R_6	1	1	0
R_7	1	1	1
External source	0	0	0

Consider the operation

$R_3 \leftarrow R_5 . R_7$

This instruction is present in the memory. It is fetched from memory and brought into the instruction decoder. Then it is analyzed or decoded. The source operands, the destination and the operation to be performed are identified. Then the control unit of the CPU takes over to complete the job as follows.

i. 101 is placed on SEL A lines. So R_5 is selected and its contents are placed on 'A' bus.

ii. 111 is placed on SEL B lines. Now R_7 will be selected and by giving suitable read/write control signal its contents are sent to ALU on B bus.

iii. Then specific combinations of operation select bits are applied to ALU. These causes ALU 10 perform $R_5 . R_7$.

iv. Then 011 is sent on SEL D lines and R_3 is selected to store result generated by ALU.

Each of the above sequence by the control unit could begin at a new clock pulse. So by the end of four clock pulses, the instruction $R_3 \leftarrow R_5 . R_7$ is completed.

Concept of Stack and its Organization:

A Stack is a storage that operates on the Last in First Out (LIFO) principle. A stack should have a stack pointer register.

The contents of this register point to the top of the stack. It is at this address that a new data can be put on and a data that can be removed from stack. The operation that are possible on a stack are insertion of data element also called a PUSH and removal of a data element also called a POP. Every PUSH or POP operation dynamically changes the contents of the stack. Apart from this there is a data register. This register temporarily holds the data that is to be copied onto stack or the data that is to be read out from the stack.

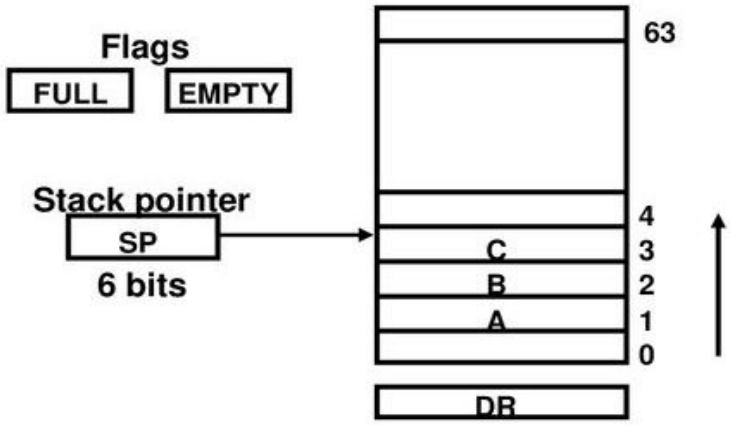
A stack can be implemented in two ways.

i. A set of general purpose registers can work as a stack by implementing LIFO principle (Register Stack).

ii. A part of the user memory can be used to create a stack (Memory Stack).

Register Stack:

A set of general purpose register can be organized to function as LIFO which is essentially the way stack works. Consider a stack having 64 GPR. The following shows the organization of such a Stack



Every general purpose register has an address starting from 0 to 63. Each register can be selected using a 6 bit address as $2^6 = 64$.

Hence the size of the SP is 6 bits. If SP = 000000; it points to register 0. If SP = 000001, it points to register 1 likewise if SP= 111111, it points to register 63. After every push the stack pointer points to the next register which is available for storing information. After every POP the stack pointer points to the register from which information has to be read out. Initially assuming stack is empty the SP points to 0. Also when stack is the EMPTY flag is set to 1.

During a PUSH, first the full flag is checked. If stack is not full, i.e., all the registers don't store the data, a new PUSH operation is possible. Then the stack pointer is incremented by 1. Into this address the new data is stored. So the first data goes in register 1. In a PUSH, first the address is incremented and then data is stored in the pointed register.

Consider SP = 63. When next PUSH is to be carried, SP becomes 0. Then the data is stored in register 0. This will be the top of stack. After every PUSH stack pointer is checked. If it is not 0 next PUSH can be carried out. In the above case, SP becomes 0. So now further PUSH is possible. All the registers have been used. So stack is full. Full flag will be set to 1. The mechanism for PUSH is as follows.

While (FULL! =1)

SP ← SP+1

M [SP] ← DR

If (SP==0) FULL ← 1, EMPTY ← 0

During a POP operation EMPTY is 0 it means stack is not empty. So a next POP operation will be possible. Now SP ← 0. When a POP is carried out the mechanism is just opposite. First the data from the register pointed by SP is read out. Then the SP will be decremented by 1. So SP becomes 63. This process continues. Now consider SP ← 1. When the next POP is to be carried out, data is read out from Register 1. Then SP is decremented to 0. After every POP operation SP is checked. If SP becomes 0 it means data has been read out from all registers. So EMPTY flag will be set to 1, else a new POP will be carried out.

Then mechanism of POP is as follows –

While (EMPTY ! =1)

DR ← M [SP]

SP ← SP-1

If (SP==0) then EMPTY= 1, FULL← 0,

Else

Go for next POP

Input Output Organization:

In a typical CPU, many I/O devices are connected. Each device has its own operating mechanism, different data formats, and different signals for operation and so on. However the CPU is basically operating on digital systems. The operations of CPU as well as data transfer speeds are very high. In such an environment, the I/O devices are communicating with the CPU. The I/O devices further could be exchanging information with CPU serially or parallely. Hence we have to have interfaces for the I/O devices.

Need for Interface:

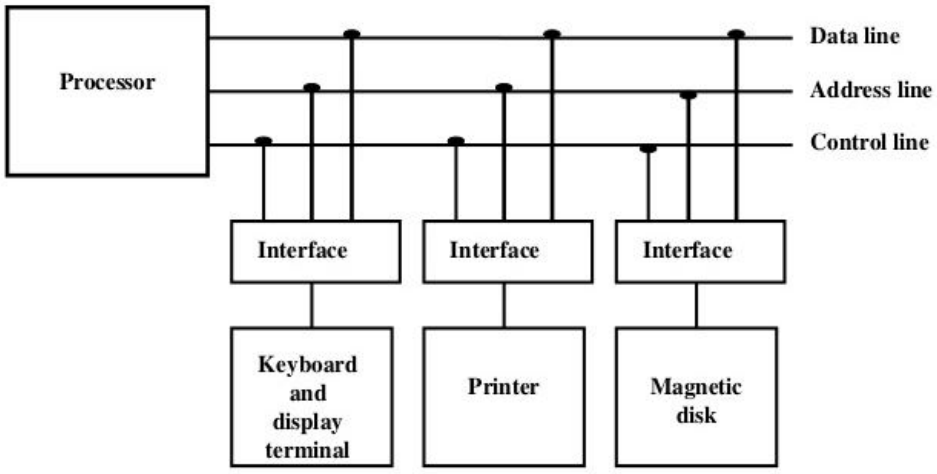
- An interface can be defined as a communication link between the CPU and I/O devices, which help in smooth transfer of data between the CPU and I/O devices.
- The interface usually a hardware. But it can be a combination of hardware and software too as in cases of device drivers.
- The interface helps resolve differences that exist between CPU and to devices. Some of the notable reasons why an interface is required are -

i. **Operation principles:** CPU is purely electronic. I/O devices are electro mechanical. Hence proper conversion of signals are necessary.

ii. **Speed:** CPU is very fast. I/O devices, on the other hand, are slow. Hence there has to be provision to store data sent by each, before they are finally sent to one another. Also additional control signals are required to ensure that all information is received correctly without loss of information. An interface can do this job.

iii. Data codes and formats of I/O devices also called peripherals are different than the word formats used by CPU and memory.

iv. Each peripheral operates differently and the operation of one should not affect the working of other. All I/O devices must independently work simultaneously.



References:

1. A text book of Basics of Computer Organization of vision publication by H.R.Arvind.
2. A text book of Basics of Computer Organization of Nirali publication by Dr.J.A.Bangali
3. www.google.com