

B.Tech.

Fourth Semester Examination

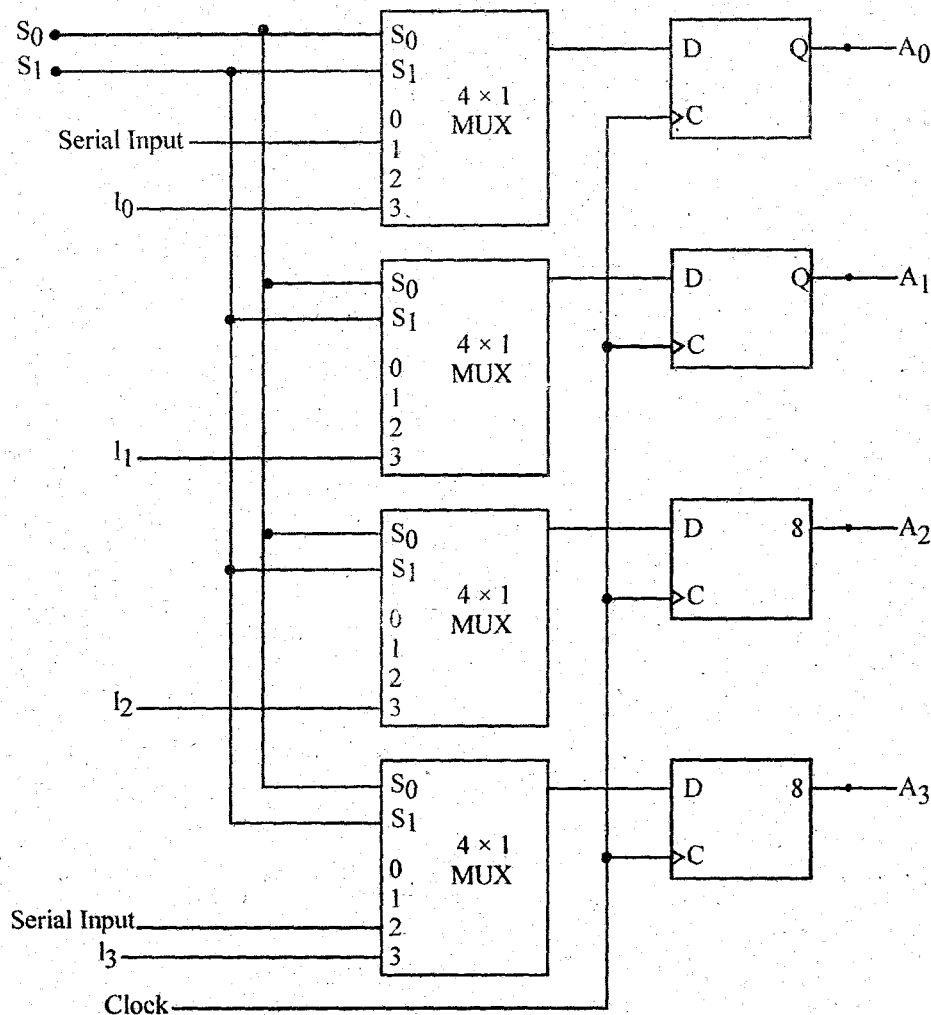
Computer Architecture & Organization (CSE-210-F)

Note : Attempt any *FIVE* questions out of given eight.

Q. 1. (a) Draw and explain bi-directional shift register.

Ans. A register capable of shifting in one direction only is called a unidirectional shift register. A register that can shift in both directions is called a bidirectional shift register.

A 4-bit bidirectional shift register with parallel load is shown in the fig. below. Each stage consists of a D flip-flop and a 4×1 multiplexer. The two selection inputs S_1 and S_0 select one of the multiplexer data inputs for the D flip-flop. The selection lines controls the mode of operation of the register according to the function table shown in fig. 2.



Q. 3. (a) Explain the addressing modes.

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

Computers use addressing mode techniques for the purpose of accommodating one or both of the following provisions :

- (i) To give programming versatility to the user by providing such facilities as pointers to memory, counters for loop control, indexing of data.
- (ii) To reduce the number of bits in the addressing field of the instruction.

Various Addressing Mode :

(i) Implied Mode : In this mode the operands are specified implicitly in the definition of the instruction.

Opcode	Mode	Address
--------	------	---------

(ii) 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.

(iii) Register Mode : In this mode the operands are in registers that reside within the CPU.

(iv) Register Indirect Mode : In this mode the instruction specifies a register in the CPU whose contents give the address of the operand rather than the operand itself.

(v) Autoincrement or Autodecrement 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.

(vi) Direct Address Mode : In this mode the effective address is equal to the address part of the instruction. The operand resides in memory & its address is given directly by the address field of the instruction.

(vii) Indirect Address Mode : In this mode the address field of the instruction gives the address where the effective address is stored in memory.

Q. 3. (b) Write down a program in 8086 assembly language to find out –ve and +ve numbers out of an given array of 10 elements.

```
Ans.  ASSUME CS:CODE, DS:DATA
      DATA SEGMENT
      LIST DW 2579H, 0A500H, 0C009H, 0159H, 0B900H
      COUNT EQU 05H
      DATA ENDS
      CODE SEGMENT
      START:  XOR BX, BX
              XOR DX, DX
              MOV AX, DATA
              MOV DS, AX
              MOV CL, COUNT
              MOV SI, OFFSET LIST
      AGAIN:  MOV AX, [SI]
              SHL AX, 01
              JC NEG
              INC BX
              JMP NEXT
      NEG:    INC DX
      NEXT:   ADD SI, 02
              DEC CL
              JNZ AGAIN
              MOV AH, 4CH
```

INT21H
CODE ENDS
END START

Q. 4. (a) What is pipelining? Explain any two different types of pipelines.

Ans. Pipelining : Pipelining is an implementation technique where multiple instructions are overlapped in execution. The computer pipeline is divided in stages.

Each stage completes a part of an instruction in parallel. Pipelining does not decrease the time for individual instruction execution. Instead, it increases instruction throughput. Throughput of the instructions pipeline is determined by how often an instruction exits the pipeline.

Types of Pipelining :

(i) Instruction pipeline

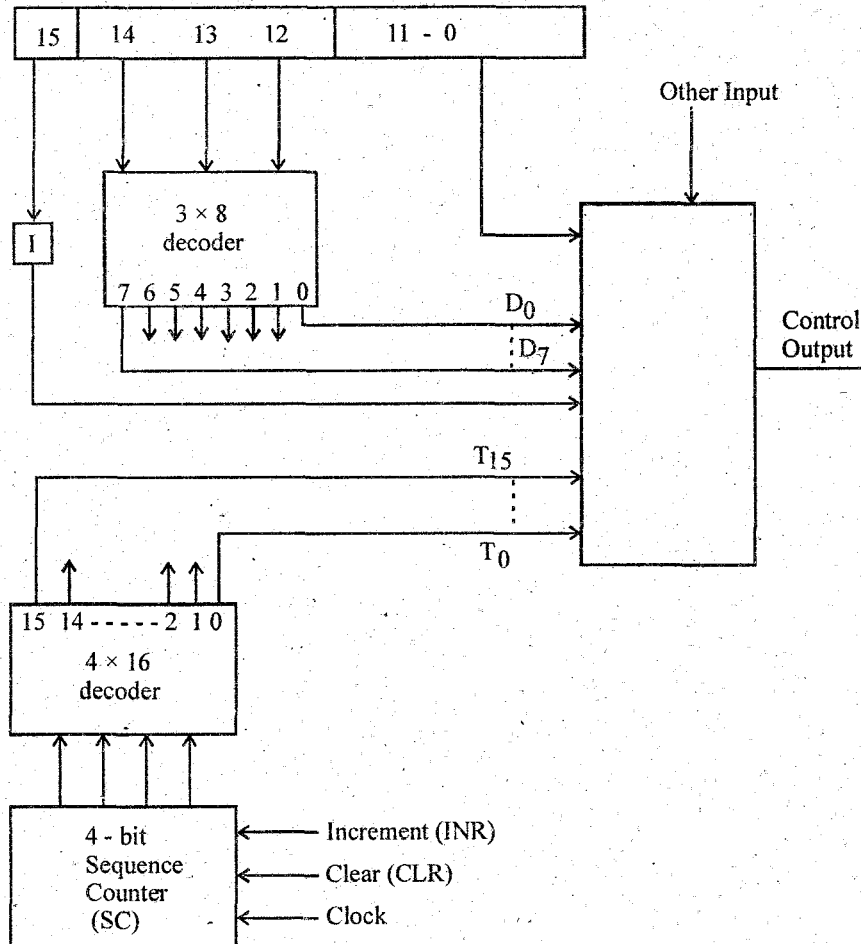
(ii) Arithmetic pipelines

Instruction pipelines are used in almost all modern day processors to a varying degree, to enhance the speed of the control unit. That is, the control unit is implemented as a pipeline consisting of several stages, where each stage is dedicated to perform one phase of the instruction cycle.

Arithmetic Pipeline : Arithmetic pipeline enhance throughput of arithmetic/logic units.

Q. 4. (b) Explain the process of implementing control unit.

Ans. Process of Implementing Control Unit :



It consists of two decoders, a sequence counter and a number of control logic gates. An instruction read from memory is placed in the instruction register (IR). The position of this register in the common bus system. The IR is divided into 3 parts. The 1 bit, the operation code and bits 0 through 11. The operation code in bits 12 through 14 are decoded with a 3×8 decoder. The eight outputs of decoder are designated by symbol D_0 through D_7 . The subscripted decimal number is equivalent to binary value of corresponding operation code. Bit 15 of instruction is transferred to flip-flop designated by symbol I. Bits 0 through 11 are applied to control logic gates. The 4-bit sequence counter can count in binary 0 through 15. The output of counter are decoded into 16 timing signals T_0 through T_{15} .

Q. 5. (a) Explain memory hierarchy. Also explain its need and importance.

Ans. Memory Hierarchy : The memory unit is an essential component in any digital computer since it is needed for storing programs & data. The memory unit that communicates directly with the CPU is called the main memory. Devices that provide backup storage are called auxiliary memory. The most common auxiliary memory devices used in computer systems are magnetic disks & tapes. They are used for storing system programs, large data files & other backup information.

Cache Memory : A special very-high speed memory called a cache is sometimes used to increase the speed of processing by making current programs & data available to the CPU at a rapid rate.

CPU logic is usually faster than main memory access time with the result that processing speed is limited primarily by the speed of main memory.

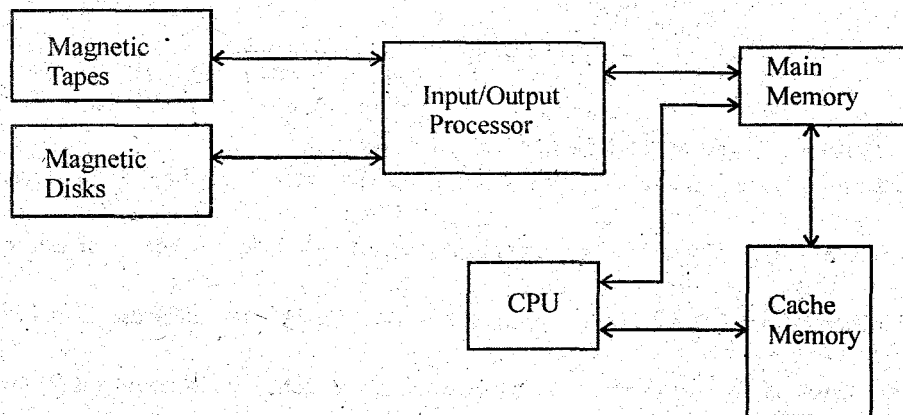


Fig. Memory Hierarchy

Q. 5. (b) Compare and contrast dynamic and static memory.

Ans. Static memory is a computer memory that contains fixed information & retains its programmed state as long as the power is on.

Static memory allocation is before run time, but the values of variables may be changed at run time static memory allocation saves running time but can't be possible in all cases.

Static memory must be declared at compile-time.

Dynamic memory can be declared at run-time using the new & delete operators (or malloc & free in C).

It is a computer memory that requires a periodic refresh to maintain its contents.

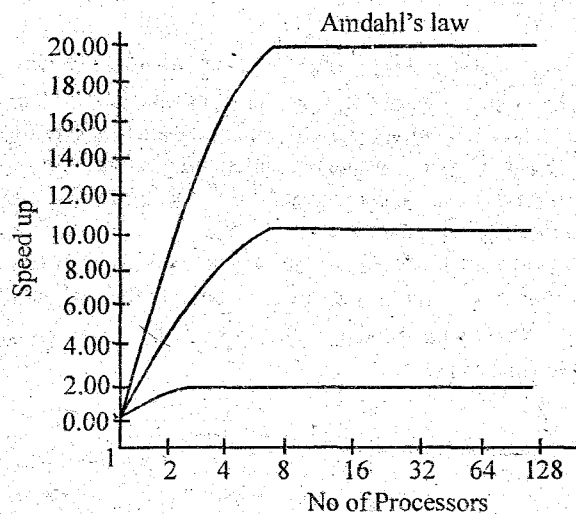
Q. 6. (a) What is Amdahl's Law? Explain.

Ans. Amdahl's Law : The performance gain that can be obtained by improving some portion of a com-

puter can be calculated using Amdahl's law. Amdahl's law states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.

Amdahl's law states that the overall speed up of applying the improvement will be

$$\frac{1}{(1-P) + \frac{P}{S}}$$



Q. 6. (b) Explain the concept of instruction level parallelism.

Ans. Instruction Level Parallelism Input : Input overlap the execution of instructions to improve performance.

The term instruction-level-parallelism refers to the degree to which, on average, the instructions of a program can be executed in parallel.

A combination of compiler-based optimization and hardware techniques can be used to maximize instruction-level-parallelism.

Instruction-level-parallelism exists when instructions in a sequence are independent & thus can be executed in parallel by overlapping.

The degree of instruction-level parallelism is determined by the frequency of true data dependencies & procedural dependencies in the code. Instruction level parallelism is also determined by what refers to as operation latency the time until the result of an instruction is available for use as an operand in a subsequent instruction.

Q. 7. (a) What is instruction cycle? Draw and explain.

Ans. Instruction Cycle :

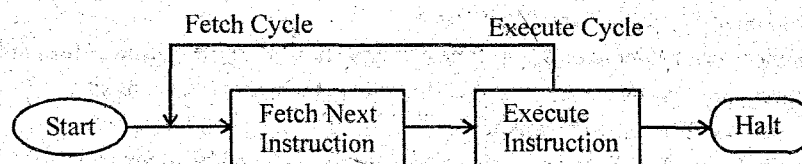


Fig. Basic Instruction Cycle

An instruction cycle (also called fetch & execute cycle, fetch-decode-execute cycle) is the time period during which a computer processes a machine language instruction from its memory or the sequence of actions that the CPU performs to execute each machine code instruction in a program.

There are typically four stages of an instruction cycle that the CPU carries out :

- (i) Fetch the instruction from memory.
- (ii) "Decode" the instruction.
- (iii) "Read the effective address" from the memory if the instruction has an indirect address.
- (iv) "Execute" the instruction.

Steps 1 & 2 are called the fetch cycle and are the same for each instruction.

Step 3 & 4 are called the execute cycle & will change with each instruction.

An instruction cycle is also called machine cycle.

Fetch & Decode : Initially, the program counter PC is loaded with the address of the first instruction in the program. The sequence counter SC is cleared to 0, providing a decoding timing signal T_0 . After each clock pulse, SC is incremented by one, so that the timing signals go through a sequence T_0 , T_1 , T_2 & so on.

Q. 7. (b) What is micro program sequence? Also explain the various types of interrupts.

Ans. Microprogram Sequence : The basic components of a microprogrammed control unit are the control memory & the circuits that select the next address. The address selection part is called a microprogram sequencer.

A microprogram sequencer can be constructed with digital functions to suit a particular application. The purpose of a microprogram sequencer is to present an address to the control memory so that a microinstruction may be read & executed. The next-address logic of the sequencer determines the specific address source to be loaded into the control address registers.

Types of Interrupts : There are three major types of interrupts that causes a break in the normal execution of a program. They can be classified as :

- (i) External interrupts
- (ii) Internal interrupts
- (iii) Software interrupts

(i) External Interrupts : External interrupts come from input-output (input/output) devices, from a timing device, from a circuit monitoring the power supply or from any other external source.

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

(iii) Software Interrupts : 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 procedure at any desired point in the program.