

B. E.

Fourth Semester Examination, Dec.-2007

COMPUTER ARCHITECTURE & ORGANIZATION

Note : Attempt only five questions.

Q. 1. (a) Explain the Flynn's classification of digital computer.

Ans. Flynn's classification :

MIMD (Multi-instruction multi-data) :

- All processors in a parallel computer can execute different instructions and operate on different data at the same time.
- Parallelism achieved by connecting multiple processors together.
- Shared or distributed memory.
- Different programs can be run simultaneously.
- Each processor can perform any operation regardless of what other processors are doing.
- Examples : Cray T90, Cray T3E, IBM-SP2

SIMD (Single-Instruction Multi-Data) :

- All processors in a parallel computer execute the same instructions but operate on different data at the same time.
- Only one program can be run at a time.
- Processors run in synchronous, lockstep function.
- Shared or distributed memory.
- Shared or distributed memory.
- Less flexible in expressing parallel algorithms, usually exploiting parallelism on array operations, e.g., F90.
- Examples : CM2, MsPar

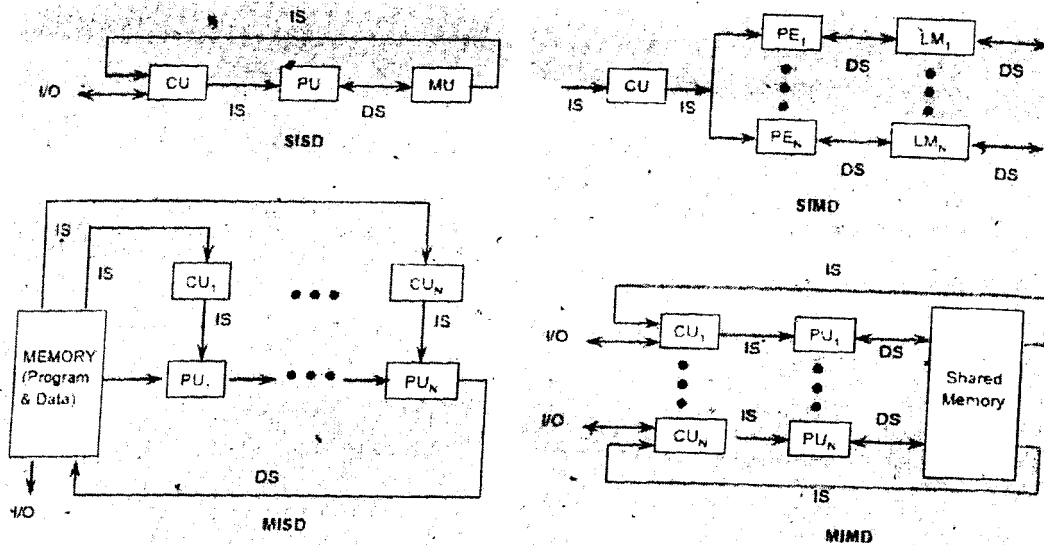
MISD (Multi-Instruction, Single-Data) :

- Special purpose computer

SISD (Single-Instruction Single-Data) :

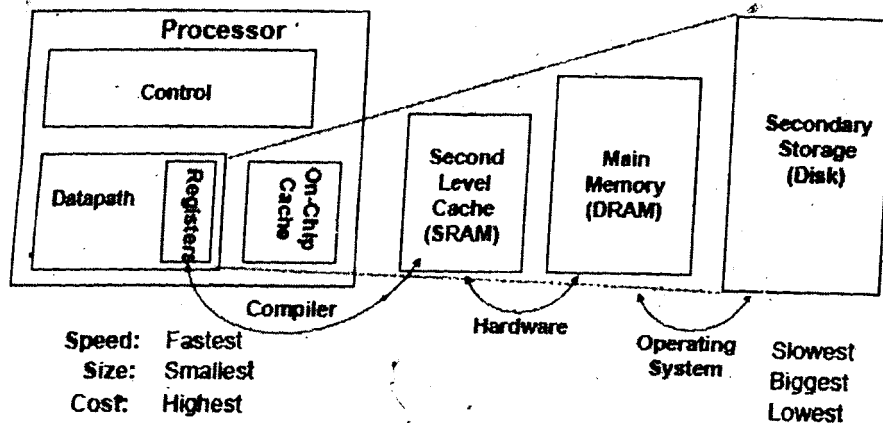
- Serial computer

CU = Control Unit PU = Processing Unit MU = Memory Unit IS = Instruction Stream
 DS = Data Stream PE = Processing Element LM = Local Memory



Q. 1. (b) Explain with diagram, the memory hierarchy in a Computer System.

Ans.



Q. 2. (a) What are the factors that effect the design of an instruction format?

Ans. A computer uses a variety of instruction code format. It is the function of the control unit within the CPU to interpret each instruction code and provide the necessary control functions needed to process the instruction. The most common fields found in instruction format are :

- An operation code field that specifies the operation to be performed.
- An address field that designates a memory address or a processor register.
- A mode field that specifies the way the operand or the effective address is determined.

Q. 2. (b) Compare the following addressing modes :

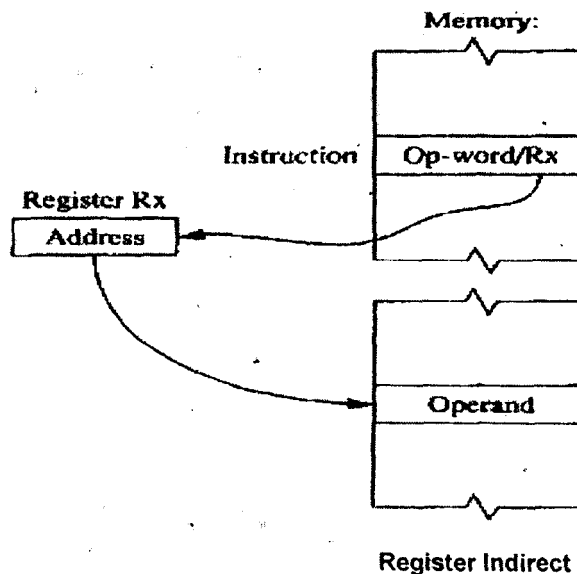
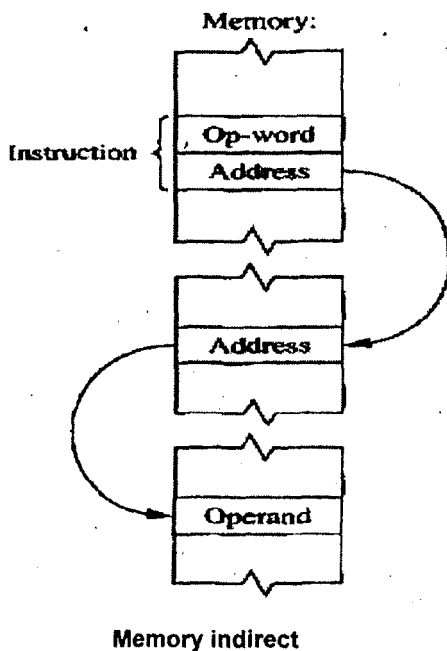
- (i) Implied & Immediate,
- (ii) Indirect & Register Indirect,
- (iii) Indexed & Base.

Ans. (i) Implied & immediate :

Implied mode : Certain instructions allow no choice of registers or location but always cause the processor to refer to the same register.

Immediate Addressing Mode : The immediate addressing mode the instruction does not state explicitly the location of the operand rather it explicitly states the operand itself. Immediate addressing is used when a particular constant value is fixed within a program itself. The value is found in memory "immediately" after the instruction code word and may never change at any time.

(ii) Indirect & register indirect :

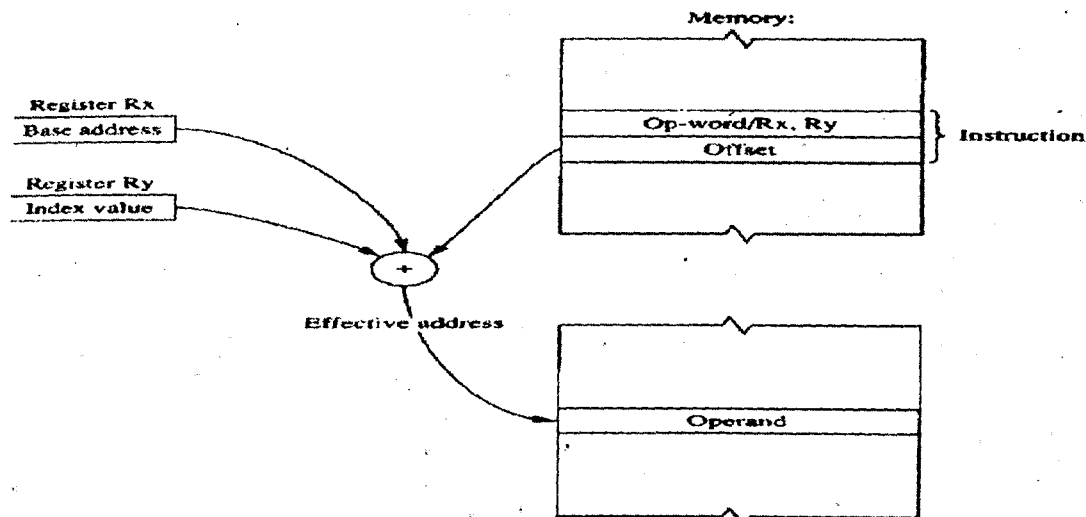


In the indirect addressing mode the instruction tells the processor neither the address of the operand nor the operand itself. Instead it tells the processor where to go to find the address of the operand. The instruction may explicit state either the address of a location in the memory or the name of the processor register, but the binary number which is found there is not the operand. Instead it is the effective address, the address of a location in the memory to which the processor must go to find the operand. The result is that the processor must take one extra step in order to locate the operand. The diagrams shows the register Indirect and Direct Addressing Mode.

(iii) Indexed & base :

Instruction which uses indexed addressing specify two registers often by coding within the opcode itself and known as "indexed addressing." During program execution the processor temporarily adds the contents of these registers to generate the effective address. One of the registers in an address register and it is said to hold the base address.

Base addressing is a similar mode wherein the instruction specifies an address register and a fixed constant. The register designation often fits within op-word and the offset usually requires post-words. In this mode the content of the register is the base and the constant is the displacement. During execution the processor adds the constant and the value in the register to generate the effective address.



Q. 3. (a) Simplify using Boolean Algebra :

$$AB'C(BD+CDE) + AC' = A(C'+B'DE).$$

Ans. Simplify using boolean algebra :

$$AB'C[BD + CDE] + AC' = A[C' + B'DE]$$

taking L.H.S.

$$AB'C[BD + CDE] + AC'$$

$$\Rightarrow \underbrace{AB'C}_{0} * \underbrace{BD}_{0} + AB'CDE + AC'$$

$$\Rightarrow AB'CDE + AC'$$

Taking R.H.S.

$$A[C' + B'DE] \Rightarrow AC' + AB'DE$$

Adding $[1 + C]$ we get,

$$AC' + AB'DE$$

Q. 3. (b) What do you mean by flip-flop? Differentiate S-R with J-k flip-flops.

Ans. Flip-flop : In digital circuits, a flip-flop is a kind of bistable multivibrator, an electronic circuit which has two stable states and thereby is capable of serving as one bit or memory. Today, the term flip-flop has come to generally denote non-transparent (clocked or edge-triggered) devices, while the simpler transparent ones are often referred to as latches.

A flip-flop is controlled by (usually) one or two control signals and/or a gate or clock signal. The output often includes the complement as well as the normal output.

J-K Flip : The JK flip-flop augments the behaviour of the SR flip-flop by interpreting the $S = R = 1$ condition as a "flip" or toggle command. Specifically, the combination $J = 1, K = 0$ is a command to set the flip-flop; the combination $J = 0, K = 1$ is a command to reset the flip-flop and the combination $J = K = 1$ is a command to toggle the flip-flop, i.e., change its output to the logical complement of its current value. Setting $J = K = 0$ does NOT result in a D flip-flop, but rather, will hold the current state. To synthesize a D flip-flop, simply set K equal to the complement of J. The JK flip-flop is therefore a universal flip-flop, because it can be configured to work as an SR flip-flop, a D flip-flop or a T flip-flop.

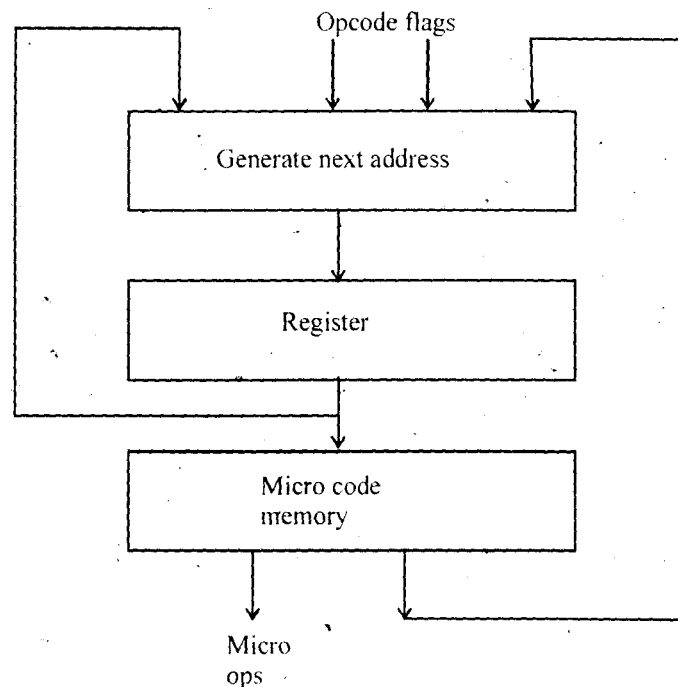
J	K	Q_{next}	Comment
0	0	Q_{prev}	hold state
0	1	0	reset
1	0	1	set
1	1	Q_{prev}	toggle

S-R FF :

The most fundamental latch is the simple SR latch (or simple SR flip-flop), where S and R stand for set and reset. It can be constructed from a pair of cross-coupled NOR (negative OR) logic gates. The stored bit is present on the output marked Q. Normally, in storage mode, the S and R inputs are both low and feedback maintains the Q and \bar{Q} outputs in a constant state, with \bar{Q} the complement of Q. If S (Set) is pulsed high while R is held low, then the Q output is forced high and stays high even after S returns low; similarly, if R (Reset) is pulsed high while S is held low, then the Q output is forced low and stays low even after R returns low.

Q. 4. (a) Write a note on microprogram sequencer?

Ans. A micro sequencer is a finite state machine. The register stores a value that corresponds to one state in CPU state diagrams. It serves the address that is input to the micro-code memory. This memory outputs a micro-instructions, the contents of memory location for that address.



Q. 4. (b) What is pipelining? Discuss how performance of a system can be enhanced with pipelining.

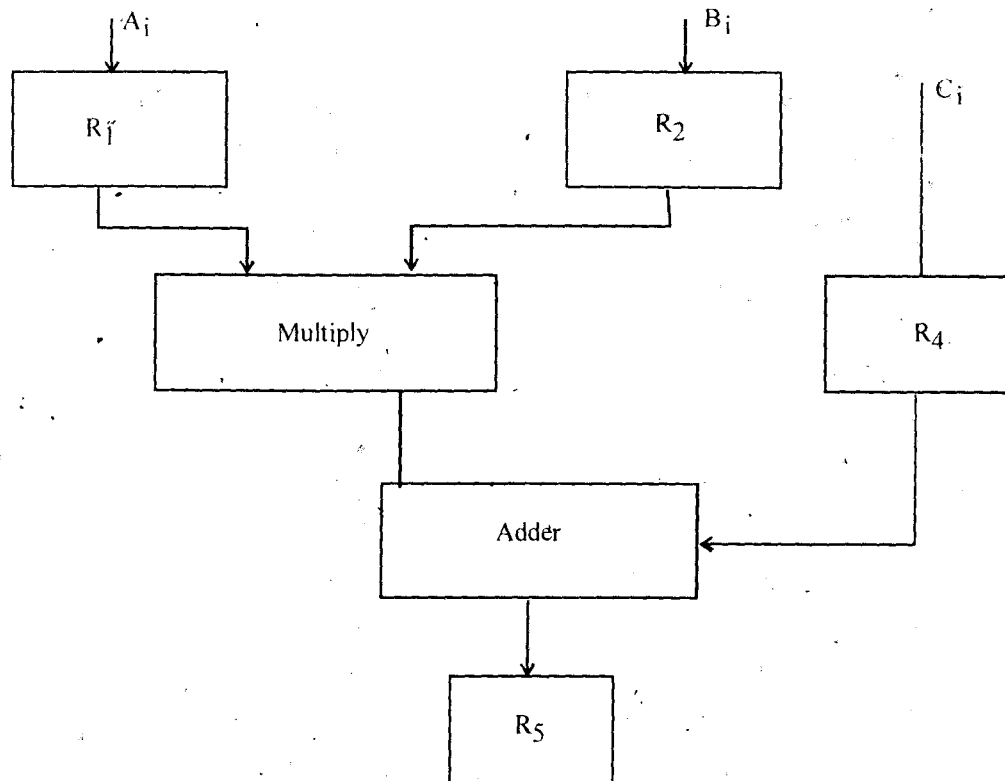
Ans. Pipelining is a general techniques for increasing processor throughput without requiring large amount of hardware. Pipelining is a technique of decomposing a sequential process into sub-process into sub-opera-

tions with each sub-process being executed in a special dedicated segment that operates concurrently with all other segments. A pipelining can be visualized as a collection of processing segments through which binary information flows. Each segment performs partial processing dedicated by the way the task is partitioned. The result obtained from computation in each segment is transferred to next segment in the pipeline. The final result is obtained after the data is passed through all the segments.

$$A_i * B_i + C_i \text{ where } i = 1, 2, \dots, 7.$$

$$R_1 \leftarrow A_i, R_2 \leftarrow B_i, R_3 \leftarrow R_1 * R_2$$

$$R_4 \leftarrow C_i, R_5 \leftarrow R_3 + R_4$$



Q. 5. (a) How many 128*8 RAM chips are needed to provide a memory capacity of 2048 bytes?

Ans. RAM chips needed $\approx \frac{2048}{128} = 16$

16 chips needed.

Q. 5. (b) How many lines of the address bus must be used to access 2048 bytes of memory? How many of the lines must be common to all chips?

Ans. $\frac{2048}{128} = 2$ lines needed.

Q. 5. (c) A computer has memory unit of 64K* 16 and a cache memory of 1K words. The cache use direct mapping with a block size of for words :

(i) How many bits are there in the Tag, Index, block & word fields of the address format?

(ii) How many blocks can the cache accommodate?

Ans. (i) Memory unit = 64K*16

Cache memory = 1 K word

$$1K = 1024 = 2^{10}$$

$$64 K = 2^{10+6} = 2^{16}$$

Handwritten calculations:
64K = 2¹⁰⁺⁶ = 2¹⁶
1K = 2¹⁰
2¹⁰ = 1024
2¹⁶ = 65536

So tag is 6 bits

Data is 10 bits.

(ii) The blocks needed for cache is 6 bits as tag field is of 6 bits.

Q. 6. Explain the concept of instruction level parallelism & process level parallelism.

Ans. Instruction level parallelism : There are at least two different kinds of parallelism in computing.

- Using multiple processors to work toward a given goal, with each processor running its own program.
- Using only a single processor to run a single program, but allowing instructions from that program to execute in parallel.

The latter is called instruction-level parallelism, or ILP.

- Instruction level parallelism (ILP) means overlapping the execution of unrelated instructions..
- Both instruction pipelining and ILP enhances instruction throughput not the execution time of the

individual instruction.

- Potential of IPL within a basic block is very limited, i.e. in "gcc" 17% of instructions are control transfer meaning on average 5 instr. + 1 branch.
- The simplest and most common way to increase the amount of parallelism is to exploit parallelism among loop iterations.

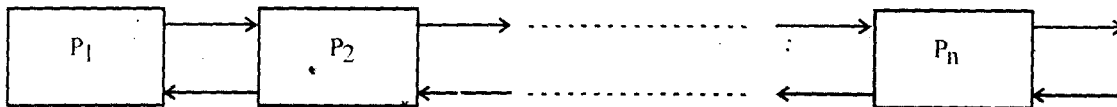
For (i = 1; i <= 1000; i++)

X[i] = x[i] + y[i];

- Techniques like loop unrolling convert loop-level parallelism into instruction-level parallelism either statically by the compiler or dynamically by hardware.
- Loop-level parallelism can also be exploited using vector processing.
- IPL feasibility is mainly hindered by data and control dependence among the basic blocks while the level of parallelism is limited by instruction latencies.

Processor level parallelism:

In this scheme large number of low cost processors work in parallel on a common task. Suppose that such a computer P(N) is connected by combining n copies of single computer P(1). If a task T can be partitioned into n subtasks of similar complexity and P(n) can be programmed so that its n processors executes the n subtasks in parallel, then we would expect Pn to process T about n times faster than P(1) can process. The advantage of processor level parallelism is tolerance of hardware and software failures. While failure of its CPU is almost always fatal to a sequential computer, a parallel computer can be designed to continue functioning perhaps at a reduced performance level, in the presence of defective CPU's. The diagram shows how number of processor work in parallel in case of processor level parallelism.



Q. 7. For 8086 microprocessor explain :

- (i) Types of interrupt,
- (ii) Set of Registers,
- (iii) Stack Organization.

Ans. (i) Types of interrupt :

There are three types of interrupts that cause a break in the normal execution of a program.

These can be classified as,

- External
- Internal
- Software.

External :

External interrupts come from I/O devices, from a timing device, from a circuit monitoring the power supply or from any other external source.

Internal :

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

Software interrupts :

These are initiated by executing an instruction. S/W interrupts is a special call instruction that behaves like an interrupt rather than a subroutine call.

(ii) Set of registers :

The registers used for 8086 are as follows :

1. Data register :

Holds memory operand.

2. Address register :

Holds address for memory.

3. Accumulator : Processor register.

4. Instruction register :

Holds instructions code.

5. Program counter :

Holds address of instruction.

6. Temporary register :

Holds temporary character.

7. Input register :

Holds input character.

8. Output register :

· Holds output character.

(iii) Stack organisation :

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 items of the stack. The two operations of a stack are insertion and detection of items. The operation of insertion is called "push" and the operation of detection is called "POP". The application of stack organisation are :

- Reverse polish notation.
- Evaluation of arithmetic expressions.

Q. 8. What do you mean by locality of reference and explain the concept of associative memory.

Ans. Locality of reference :

Memory hierarchy was developed based on program behaviour known as locality of reference. Patters on and Hennessary pointed out 90-10 rule which states that a typical program may spend 90% of its execution time and 10% of code. There are 3 dimensions for locality of reference :

- Temporal
- Spatial
- Sequential.

Temporal :

Recently referenced items are likely to be referenced again in near future. e.g., program construct, iterative loops, process stack, temporary variable.

Spatial :

It refers to the tendency for a process to access addresses near one another. e.g., operations on table, arrays involves access of certain clusters and areas in address space.

Sequential :

In typical programs, the executions of instructions follows a sequential order or the program order, unless a branch instruction out-order-execution.

Associative memory :

The search procedure is a strategy for choosing a sequence of addresses, reading the content of memory at each address and comparing. The information read with the item being reached until a match occur. The time required to find an item stored in memory can be reduced if stored data can be identified for access by the content of the data itself rather than by any address. A memory unit accessed by contents is called an associative memory or content addressable memory (CAM). It is more expensive than RAM because each cell must have storage capability as well as logic circuit for matching its contents with an external argument. Block diagram of associative memory is shown below :

