B.E.

Seventh Semester Examination, 2010-2011

Advanced Computer Architecture (CSE-401-E)

Note : Attempt any five questions. All questions carry equal marks.

Q. 1. (a) What are different basic data types.

Ans. Data Types : There are some major data types :

(i) Integers, long and short.

(ii) Integer, signed and unsigned.

(iii) Char, signed and unsigned.

(iv) Floats and doubles.

(i) Integers, Long and Short : An integer constant depends opoun the compiler. For a 6 bit compiler like turbo C or turbo C++ the range is –32768 to 32767.

For a 32-bit compiler the range would be –2147483648 to +2147483647 short and long integers would usually occupy two and four bytes respectively.

(ii) Integer Singed and Unsigned : Sometimes, we know in advance that the value stored in a given integer variable will be always positive.

When it is being used only count things, for example,

```
unsigned int num_student;
```

In such a case we can declare the variable to be unsigned.

With such a declaration, the range of permissible integer values (for a 16-bit OS) will shift from the range –32768 to +32767 to the range 0 to 65535. Thus, declaring an integer as unsigned almost doubles the size of the largest possible value that it can otherwise take.

(iii) Chars, Signed and Unsigned : Parallel to signed and unsigned ints (either short or long), there also exist signed and unsigned chars. Both occupy one byte each, but having different ranges.

To begin with, it might appear strange as to how a char can have a sign.

A signed char is same as an ordinary char and has a range from –128 to +127; whereas an unsigned char has a range from 0 to 255.

(iv) Floats and Double : A floats occupies four bytes in memory and can range from –3.4e38 to +2.4e38. If this is unsufficient than C offers a double data type that occupies 8 bytes in memory and has a range from –1.7e308 to +1.7e308.

The essence of all the data types that we have learnt so far has been captured in figure 1.1

Data Type	Range	Bytes	Format
Signed char	–128 to +127	1	%c
Unsigned char	0 to 255	1	%c
Short signed int	–32768 to +32767	2	%d
Short unsigned int	0 to +65535	2	%u
Signed init	–32768 to +32767	2	%d

Unsigned init	0 to 65535	2	%u
Long signed init	–2147483648 to +2147483647	4	% ld
Long unsigned int	0 to 4294967295	4	%ld
Float	–3.4e38 to +2.4e38	4	%f
Double	–1.7e308 to +1.7e308	8	% lf
Dong double	–1.7e4932 to 1.7e4932	10	%lf

Q. 1. (b) What do you mean by interpretation. Give its advantages and disadvantages.

Ans. The Machine Interpretation : Management of the interpretation process is the responsibility of the decoder (a part of the implementation mechanism). The process of interpreting or executing on instruction begin & with the decoding of the opcode field from the instruction. The decoder activates storage and registers for a series of state transitions that correspond to the action of the opcode. The image machine storage consists of registers and memory, which are both explicitly stated in the instruction and implicitly defined by the instruction.

Explicit Registers Include :

(i) General purpose register (GPR)

(ii) Accumulators (ACC)

(iii) Address Registers.

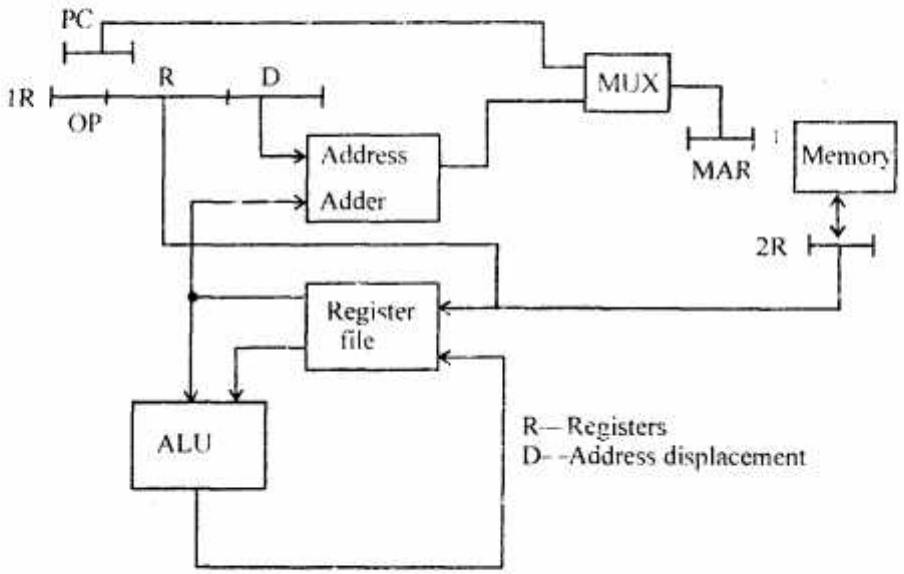


Fig. Some Processor Registers and Data Paths

Implicit registers consist of the following (fig.)

(i) Program or instruction counter (PC) most instruction formates imply the next instruction in sequence

as the current location plus the length of the current instruction.

(ii) Instruction Register (IR) : The register holds the instruction being interpreted or executed. Decoding is performed on the opcode held in the register

(iii) Memory Address Register (MAR) : This is the register for a memory operation.

(iv) Storage Register (SR) : This is sometimes referred to as the memory buffer register and contains the data used in the memory (to or from) operation.

(v) Special Use Registers : Usage depending on instructions.

Data paths connect the output of one register to the input of another and may include combinational logic.

The opcode generally defines which of the many data paths are used in execution. The collection of all opcodes defines the data paths required by a specific Architecture.

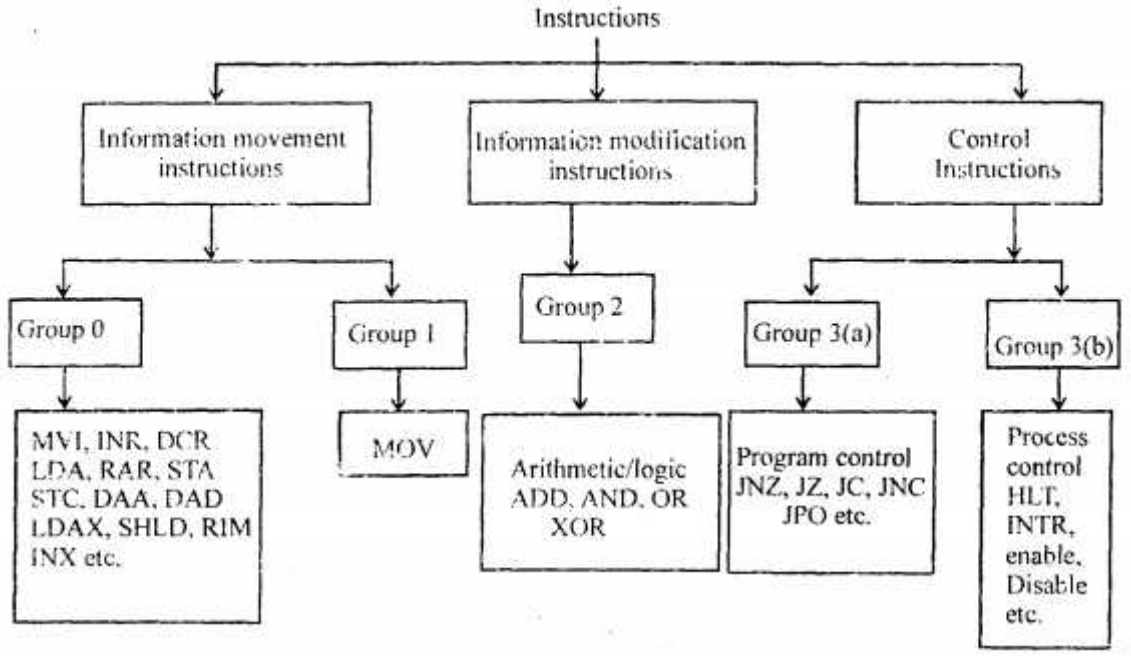
A register may be connected to multiple output destination registers and accept input from one of several source registers in any one cycle.

A register output is gated to various destinations with which it can communicate in a single cycle. The Activation of a particular data path is done through a control point.

Q. 2. (a) Explain various instructions of a microprocessor.

Ans. A microprocessor instruction is similar to the instruction given to a human being to perform a specific task. It is the binary pattern consisting of '1' and 0 designed by manufacturer of the particular microprocessor to perform a specific task (function). The 8085 micprocessor includes all instructions of the 8085 plus 2 additional instructions i.e., RIM and SIM. Thus, in all the 8085 has 74 instructions called instruction set.

The whole instructions of the microprocessor can be divided into 3-general headings as depicted in fig.



Both information movement instructions and tructions are further sub-divided in two groups. Information movement instruction contain group 0 whereas control instruction includes group 3(a) and 3(b). Thus, whole instruction set of 8085 is di four group, namely group 0, group 1, group 2, group 3.

Information Movement Instructions :

Group 0 : Group 0 covers the following type

(i) Immediate data movement.

(ii) Incrementing and decrementing of regis ster pairs.

(iii) Loading and storing of registers and reg

(iv) Shifting (Rotate) and complementing of s of accumulator.

Group 1 : Group 1 is the largest group of instru is called MOV group. Group 1 only moves data/ information from one source to destination without source data i.e., it copies the source data to the destination.

Information Modification Instructions :

Group 2 : The contents of certain registers m ified by these instructions. Modification is the accomplished by performing either arithmetic or lo; ons on the contents of the register.

The arithmetic operation & for 8085 are :

(i) Addition

(ii) Subtraction

Control Instructions : Group 3(a) or program control operation. As per the heading, the program control operation transfers the execution of a program from its present location to some other location in the memory. Transfer may be conditional based on the contents of the status registers or unconditional without restrictions. Furthermore, each transfer may or may not require return transfer. In return transfer the present address of execution is saved with the help of the stack pointer (SP) so that the control can return to the place where it left the program. Non-returning transfer does not require saving the present address.

Group 3 (b) or Processor Control Operations : The process control instructions do not require any operand. It perform operations directly on the microprocessor and are a few in numbers.

Typical instructions in this group are processor Hault, Enable/Disable. Interrupts etc.

Q. 2. (b) Discuss the professor Evaluation Matrix.

Ans. Multithreaded Von Neumann Architecture can be traced back to the CDC 6600 manufactured in the mind 1960's.

Multiple function units in the 6600 CPU can execute different operations simultaneously using a scorboarding control.

The very first multithreaded multiprocessor was the denelcor HEP designed by burton smith in 1978. The HEP was built with 16 processors driven by a 10-MHz clock, and each processor can HEP was built with 16 processors driven by a 10-MHz clock, and each processor can execute 128 threads (called processes in HEP terminology) simultaneously.

We describe the tera architecture to understand processor evaluation matrix, its processors and thread state and the tagged memory/registers.

The unique features of the tera include not only the high degree of multithreading but also the explicit-dependency lookhead and the high degree of super pipelining in its processor-network memory operations. These advanced features are mutually supportive. The first tera machine are expected to appear in late 1993.

Let I_C be the number of instructions in a given program or the instruction count. The CPU time (T is second/program) needed to execute the programs is estimated by finding the product of three contributing factors.

$$T = I_C \times CPI \times \tau \quad ...(1)$$

The CPI of an instruction type can be divided into two component terms corresponding to the total processor cycles and memory cycles needed to complete the execution of the instruction depending on the instruction type the complete instruction cycle may involve one to for memory references (one for instruction fetch, two for operand fetch and one for store results). Therefore we can rewrite eq. 1 as follows,

$$T = I_C \times (P + m \times k) \times \tau$$

Where P is the number of processor cycles needed for the instruction decode and execution, m is the number of memory references needed, k is the ratio between memory, cycle and processor cycle. I_C is he instruction count and τ is the processor cycle time.

Q. 3. (a) What do you mean by virtual to real translation.

Ans. The main memory is considered the physical memory in which many programs want to reside. However the limited size physical memory cannot load in all programs simultaneously. The virtual memory concept was introduce to alleviate this problem. The idea is to expand the use of physical memory among many programs with the help of an auxiliary (backup) memory such as disk arrays only active programs or portions of than become residents of the physical memory at one time. The vast majority of programs or inactive programs are stored on disk.

All programs can be loaded in and out of the physical memory dynamically under the coordination of the operating system. To the users virtual memory provides them with almost unbounded memory space to work with without virdual memory, it would have been impossible to develop the multiprogrammed or time sharing computer systems that are in use today.

Address Spaces : Each word in the physical memory is identified by a unique physical address. All memory words in the main memory form a physical address space. Virtual addresses are generated by the processor during compile time. In UNIX systems each process created is given a virtual address space which contains all virtual addresses generated by the compiler.

The virtual addresses must be translated in to physical addresses at run time. A system of translation tables and mapping functions are used in this process. The address translation and memory management policies are affected by the virtual memory model used and by the organization of the disk arrays and of the main memory.

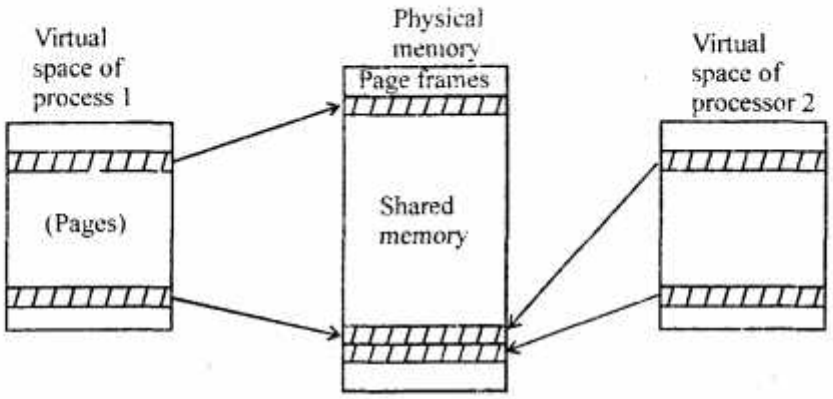
Address Mapping : Let V be the set of virtual address generated by a program (or by a software process) running on a processor. Let M be the set of physical address allocated to run this program. A virtual memory system demands an automatic mechanism to implement the following mapping :

$$f_t : V \rightarrow M \cup \{\phi\}$$

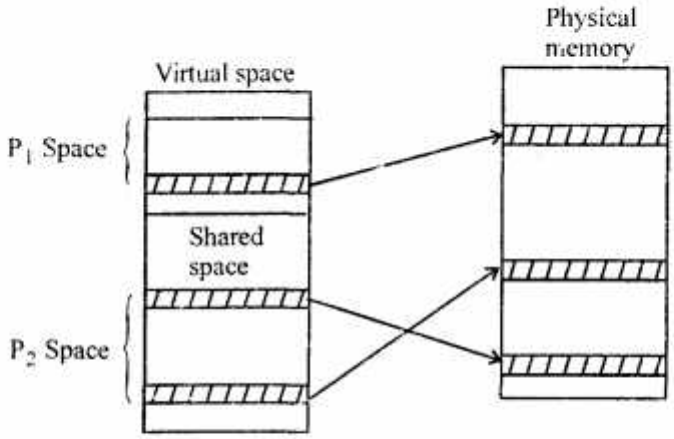
This mapping is a time function which varies from time to time because the physical memory is dynami-

cally allocated and deallocated consider any virtual address $v \in V$.

Virtual to Real Translation : The processes demands the translation of virtual address in to physical address various schemes for virtual address translation are sumcrised in figure (a). The translation demands the use of translation maps which can be implement in various ways.



(a) Private Virtual Memory Spaces in Different processors.



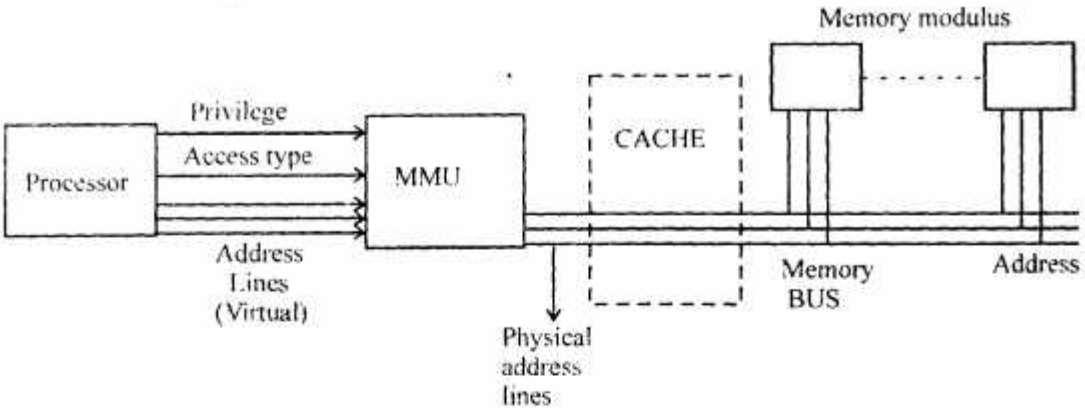
(b) Globally Shared Virtual Memory Space

Translation maps are stored in the cache, in associative memory, or in the main memory. To access these maps a mapping function is applied to the virtual address.

Q. 3. (b) Explain the organization of Cache.

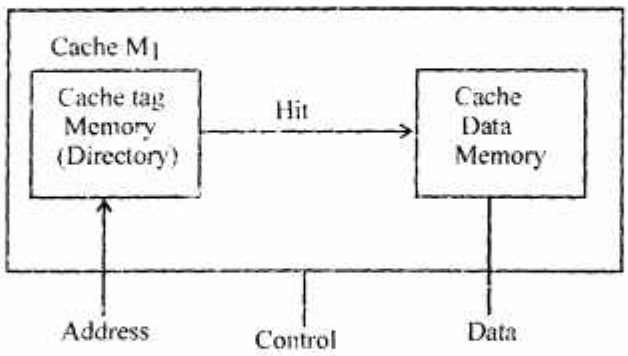
Ans. A cache memory may be inserted between the MMU (if there is one) and the physical main memory to give the processor the appearance of a faster main memory. The cache serves as the fastest, most expensive level of the memory hierarchy; it is inserted to decrease the average main memory access time as seen from the processor.

Following fig. 3.1 show the location with main memory.



(Location of Cache)

Fig. 3.2 shows the cache organization and the principal components of cache memory words are stored in a cache data memory and are grouped in to small pages called cache blocks or lines.



(Fig. 3.2 Basic Structure of Cache)

The contents of cache's data memory are thus copies of a set of main memory blocks. Each cache block is marked with its block address, referred to as a tag. So the cache knows to what part of memory space the block belongs.

The collection of tag address currently assigned to the cache, which can be non-continguious is stored in a special memory, the cache tag memory or directory.

Two general ways of introducing a cache into a computer appear in (Figure 3.3).

In the look-aside design of figure 3.3 a the cache and main memory are directly connected to the system bus. In this design the CPU initiates a memory access by placing a (real) Address A_i on the memory address bus at the start of a read (load) or write (store) cycle. The cache M_1 immediately compares A_i to the tag address currently residing in its tag memory. If a match is found in M_1 that is a cache hit occurs, the access is completed by a read or write operation execute in the cache; main memory M_2 is not involved. If no match with A_i is found in the cache that is a cache miss occurs, then the desired access is completed by a real or write

operation directed to M_2.

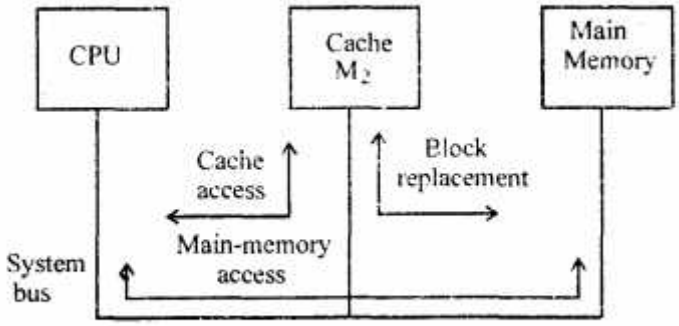
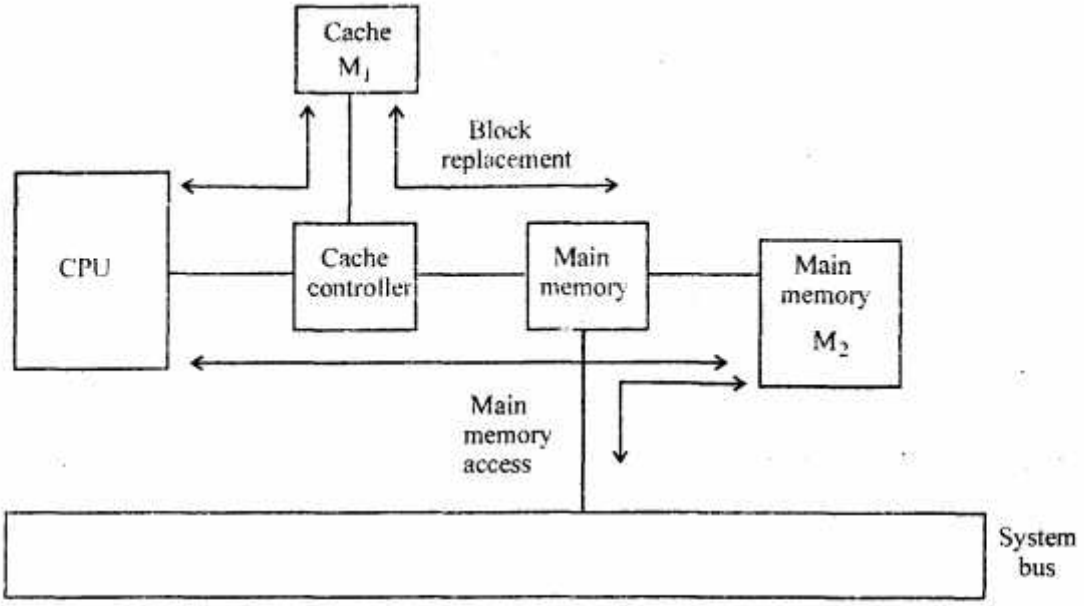


Fig. (a)



A faster, but more costly organization called a look through cache appears in figure 3.3 (b). The CPU communicates with the cache via a separate (local) bus that is isolated from the main system bus. The system bus is available for use by other units, such as IO controllers, to communicate with main memory.

Q. 4. (a) Discuss the processor memory modeling using queuing theory.

Ans. Processor Memory Modeling Using Queueing Theory : Simple processors are unbuffered; when a response is delayed due to conflict this delay directly affects processor performance by the same amounts.

More sophisticated processors including certainly almost pipelined processors-make buffered requests to memory, unless a cache is used even with pipelined processors that use cache. Some of request (such as writes) may be buffered. Whenever request are buffered, the effect of contention and the resulting delay are reduced. The simple models fail to accurately represent the process-memory relationship. More powerful tools

that incorporate buffered requests are needed.

For These Tools, We Turns to Queuing Theory : Open and close queue models are frequently used in the evaluation of computer system designs.

In this we review and present the main results for simple queuing system without derivation of the underlying basic queue equation. While queuing models are useful in understanding memory behaviour, they also provide a robust basis to study various computer system interactions such as a multiple processor and input/output.

Suppose requestors desire service from a common server. These requestors are assumed to be independent from one another, except that they make a request on the basis of probability distribution function called the request distribution function.

Similarly the server is able to processor request one at a time, each independently of the others, except that the service time is distributed. According to server probability distribution function. The mean of arrival or request rate is measured in items per unit of time and is called X and the mean of service rate (e) defines a very important parameter in queueing systems called the utilization or the occupancy.

The higher the occupancy ratio, the more likely it is that requestors will be waiting (in a buffer or queue) for service and the longer the expected queue length of items awaiting service.

In open queuing systems if the arrival rate equals or exceeds the service rate an infinitely long queue develops infront of the server, as requests are arriving faster than or at the same rate at which they can be served.

Queueing Theory : As processor have become increasingly complex, their performance can be predicted only statistically.

Indeed, often times running the same job on the same processor on two different occasions may create significantly different execution times (based perhaps on initial state of the list of available pages maintained by the operating system).

Queueing theory has been a powerful tool in evaluating the performance of not only memory systems, but also input/output systems, networks and multiprocessor system.

Q. 4. (b) Discuss the difference between various queuing models.

Ans. Open, Closed and Mixed Queue Models : Open-queue models are the simplest queuing form. These models (at least as used here) assume :

1. Arrival rate independent of service rate.
2. As a consequence of (1) a queue of unbounded length as well as (potentially) unbounded waiting time.

Many will be recognize the suitability of open queue models to contain highway congestion or bridge access situations, but will also recognize the unsuitability to computer system. In a processor memory interaction, the processors rate decreases as memory congestion increases. The arrival rate is function of the total service time (including waiting time). This latter type of situation can be modeled by a queue with feedback. The system is initially offered a request rate (λ_o), but certain requests cannot immediately enter the server and are held in a queue. The requestor slows down to accommodate this and the arrival rate is now λ_a see in figure.

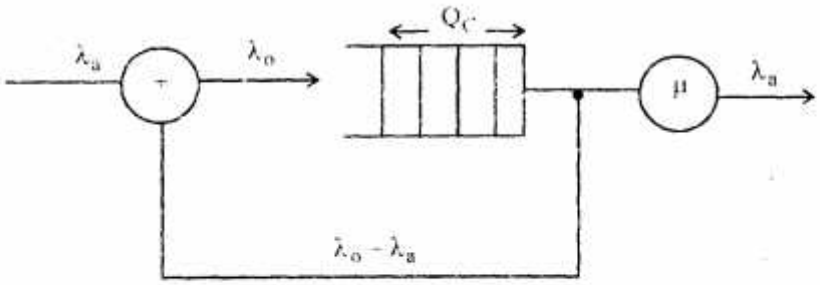


Fig (Capacity Queues)

We call such systems closed queue (or a capacity queue) and designate them Q_C. These queue usually have a bounded size and waiting time.

It is also possible for systems to behave as open queueing systems upto a certain queue size, than they behave as closed queue.

We call such systems mixed queue systems. The assumptions in developing the open queue might make it seem as unlikely candidate for memory systems modeling, yet its simplicity is attractive and it remains a useful first approximation to memory systems behaviour.

Q. 5. (a) Explain vector and scalar balance point.

Ans. In a super computer, separate hardware resources with different speeds are dedicated to concurrent vector and scalar operations. Scalar processing is indispensable for general-purpose architecture vector processing is needed for regularly structured parallelism in scientific and engineering computations. These two types of computations must be balanced

The Vector Scalar Balance Point is Defined as the Percentage of vector code in a program required to achieve equal utilization of vector and scalar hardware. In other words, we expect equal time spent in vector and scalar hardware so that no resources will be idle.

If a system is capable of 9 M flops in vector mode if the code is 90% vector and 10% scalar, resulting in a vector balance point of 0.9.

If way not be optimal for a system to spend equal time in vector and scalar modes.

However the vector scalar balance point should be maintained sufficiently high, matching, the level of vectorization in user programs.

Vector performance can be enhanced with replicated functional unit pipelines in each processor. Another approach is to apply superpipelining on vector units with a double or triple clock rate with respect to scalar pipeline operations longer vectors are required to really achieve the target performance. It was projected that by the year 2000, an 8 G flops peak will be achievable with multiple functional units running simultaneously is a processor.

Q. 5. (b) What are multiple issue machines.

Ans. Multiple Issue Machines : In a superpipelined architecture a deeper instruction pipelining is employed where the basic pipeline cycle time is a fraction of the base machine cycle time. During machine cycle several instructions can be issued in to the pipeline. In a superpipelined machine of degree m, the instruction parallelism capable of being exploits is m superpipelined and super scalar machines are considered the dual of each other although they may have different design trade-offs.

Very Long Instruction World (VLIW) architecture exploit parallelism by packaging several basic machine

operations in a single instruction word.

These basic operation consist of simple load/store memory operations and register to register ALU operations.

Each operation is simple in that it can be issued in a single clock cycle but may take several cycles to complete.

VLIW architecture make extensive use of compiler techniques to defect parallelism and package them in to long instruction word.

Q. 6. (a) Discuss partitioning in multiprocessors.

Ans. Partitioning in Multiprocessors : The goal of parallel processing is to exploit parallelism as much as possible with the lowest overhead.

"Program partition in multiprocessor is a technique for decomposing a large program and data set into many small pieces for parallel execution by multiple processors."

Program partitioning involves both programmers and the compiler. Parallelism detection by user is often explicitly expressed with parallel language constructs. Program reconstructing techniques can be used to transform sequential program in to a parallel form more suitable for multiprocessors.

Ideally, this transformation should be carried out automatically by a compiler.

Program replication refers to duplication of the same program code for parallel execution on multiple processors over different data set. Partitioning is often practiced on a shared memory multiprocessor system, while replication is more suitable for distributed memory message passing multicomputers.

So, far only special program constructs, such as independents loops and independent scalar operations have been successfully parallelised.

Clustering of independent scalar operations in to vector or VLIW instructions is another approach toward this end. "Program partition determines whether a given program can be partitioned or split into pieces that can execute in parallel or follow a certain prespecified order of execution. Some programs are inherently sequential in nature and thus cannot be decomposed into parallel branches. The detection of parallelism in programs requires a check of the various dependency relations.

Q. 6. (b) Explain memory coherence in shared memory multiprocessors.

Ans. Memory Coherence : The coherence property requires that copies of the same information item at successive memory level be consistent. If a word is modified in the cache, copies of that word must be updated immediately or eventually at all higher levels. The hierarchy should be maintained as such. Frequently used information is often found in the lower levels in order to minimize the effective access time of the memory hierarchy. In general there are two strategies for maintaining the coherence in a memory hierarchy.

The first method is called write through (WT), which demands immediate update in M_{i+1} if a word is modified in M_i, for i = 1, 2, ... n–1.

The second method is write back (WB), which delays the update in M_{i+1} until the word being modified in M_i is replaced or removed from M_i. The performance of a memory hierarchy is determined by the effective access time T_{eff} to any level in the hierarchy. It depends on the hit ratio and frequencies of successive level.

(i) Hit Ratios : Hit ratio is a concept defined for any two adjacent levels of a memory hierarchy.

(ii) Effective Access Time : In practice, we wish to achieve as high a hit ratio as possible at M_1. Every time

a miss occurs, a penalty must be paid to access the next higher level of memory.

Q. 7. (a) Discuss the evolution of computer architecture.

Ans. The study of computer architecture involves both hardware organization and programming/software requirement. As seen by an assembly language programmer.

Computer architecture is abstracted by its instruction set, which includes opcodes (operation code), addressing modes, registers, virtual memory etc.

From the hardware implementation point of view, the abstract machine is organised with CPU's, caches, buses, microcode, pipelines, physical memory etc.

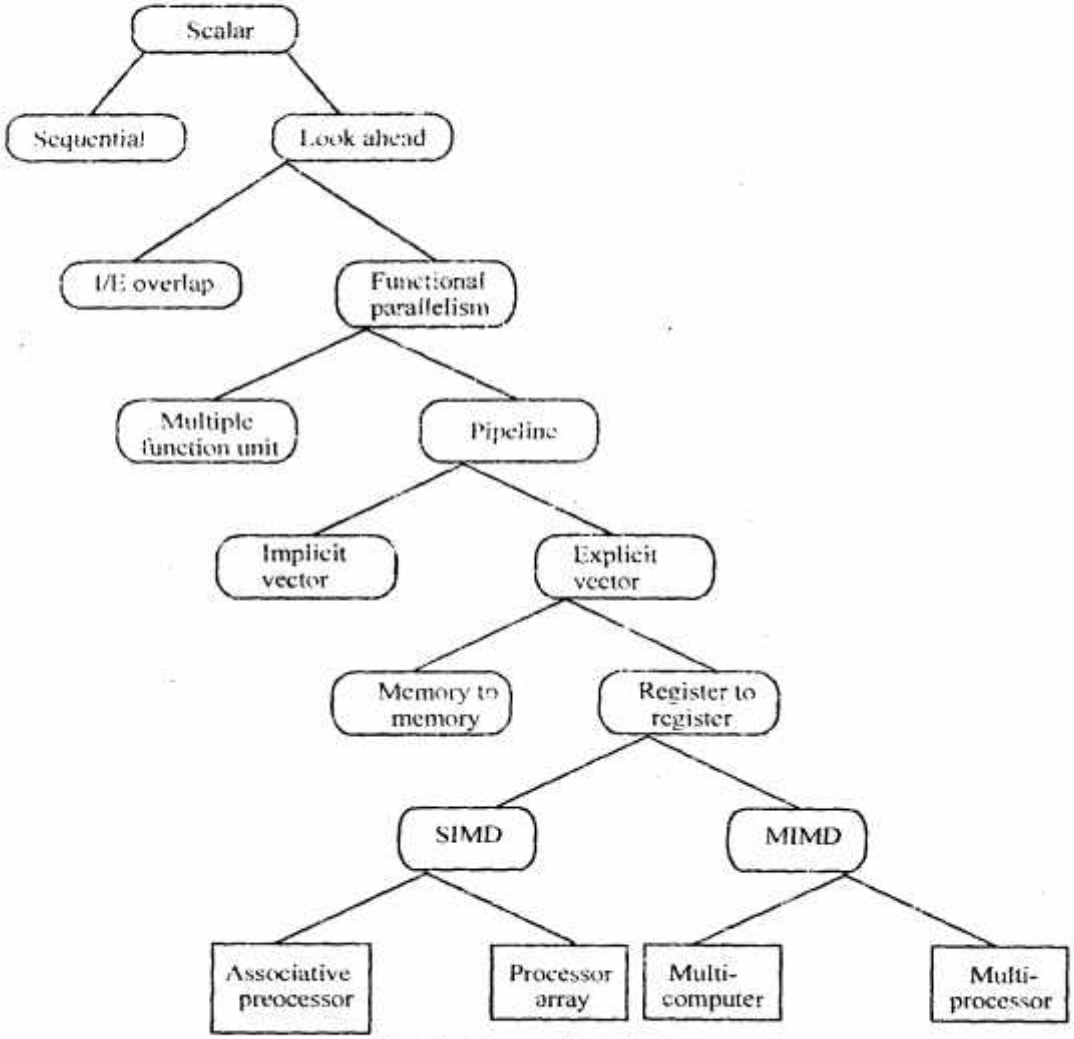


Fig. (Architectural Evolution)

Over the past four decades, computer architecture has gone through evolution rather than revolution changes.

Sustaining features are those that were proven performance deliverers.

As depict in fig. we started with the Von Neumann architecture built as a sequential machine executing scalar data.

The sequential computer was improved from bit-serial to word-parallel operations and from fixed point to floating point operations. The Von Neumann Architecture is slow due to sequential execution of instructions in programs.

Look, Parallelism and Pipelining : Lookhead techniques were introduced to prefetch instructions in order to overlap I/E operations and to enable functional parallelism. Functional parallelism was supported by two approaches. One is to use multiple functional units. Simultaneously and the other is to practice pipelining at various processing levels.

The latter includes pipelines instruction execution, pipelined arithmetic computations and memory Access operations. Pipelining has proven especially attractive in performing identical operations repeatedly over vector data strings. Vector operations were originally carried out implicitly by software controlled looping using scalar pipeline processor.

Q. 7. (b) What are cache references per instruction.

Ans. When a cache is indexed or tagged with virtual address it is called virtual address cache.

The physical address generated by the MMV can be saved in tags for later write back but it is not used during the cache lookup operations.

The virtual address cache is motivated with its enhanced efficiency to access the cache faster, overlapping with the MMV translations.

The major problem associated with a virtual address cache is aliasing. When different logically addressed data have the same index/tag in the cache. Multiple process may use the same range of virtual addresses.

This aliasing problem may create confusion of two or more processes access the some physical cache location. One way to solve the aliasing problem is to flush the entire cache whenever aliasing occurs.

Large amount of flushing may result in a poor cache performance, with a low hit ratio and too much time wasted in flushing. When a virtual address cache is used with UNIX, flushing is needed after each control switching.

Before input output writes or after input output read, the cache must be flushed further more aliasing between the unix kernal and a data is a serious problem. All of these problems will introduce additional system overhead.

The instruction scheduler exploits the pipeline hardware by filling the instructions. The instruction set of computer specifies the primitive commands or machine instructions that a programmer can use in programming the machine. The complexity of an instruction set is attributed to the instruction formats, data formats. Addressing modes general purpose registers, opcode specification and flow control mechanism used. Based on past experience in processor design, two schools of thought on instruction set architectures have evolved namely, CISC & RISC.

Q. 8. Write short notes on :

(i) Waiting time

(ii) Multiple issue machines

(iii) Synchronization in multiprocessor.

Ans. (i) Waiting Time : It is the sum of time intervals for which the process has to wait in the ready queue. The CPU scheduling algorithm should try to make this time as less as possible for a given process.

For a given instruction set, we can calculate an average CP & over all instruction types, provided we know their frequencies of appearance in the program. An accurate estimate of the average CPI requires a large amount of program code to be traced over a long period of time.

(ii) Multiple Issue Machines : In a superpipelined architecture a deeper instruction pipelining is employed where the basic pipeline cycle time is a fraction of the base machine cycle time, during machine cycle several instructions can be issued into the pipe line. In a superpipeline machine of degree the instruction parallelism capable of being exploites is m superpiplined and superscalar machines are considered the dual of each other although they may have different design trade offs.

(iii) Synchronization in Multiprocessor : Synchronization of data parallel operations is done at compile time rather than at run time.

Hardware synchronization is enforced by the control unit to carry out the locked execution of SIMD programs.

We addressed below instruction/data broadcast masking, data routing operations & separately. Languages, compiler and the conversion of SIMD program to run on MIMD multicomputer are also discussed.