Module-2

3 a. What are the characteristic of CISC and RISC architecture?
b. What are the virtual memory models for multiprocessor system?
c. Explain address translation mechanism using TLB and page table.

OR

4 a. Explain typical superscalar RISC processor architecture.
b. Explain inclusion, coherence and locality properties.

Table 4.1 *Characteristics of Typical CISC and RISC Architectures*

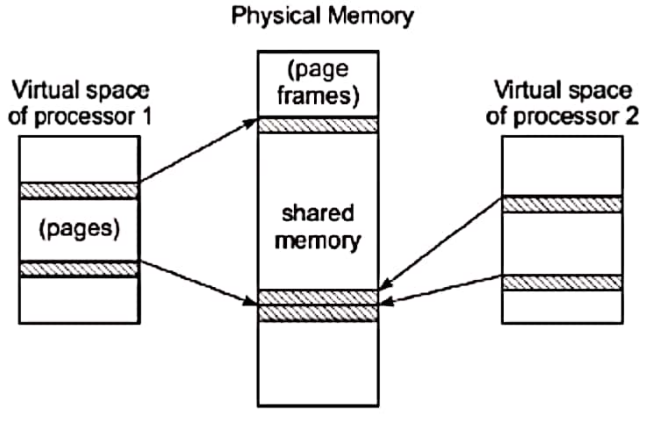
Architectural Characteristic	*Complex Instruction Set Computer (CISC)*	*Reduced Instruction Set Computer (RISC)*
Instruction-set size and instruction formats	Large set of instructions with variable formats (16–64 bits per instruction).	Small set of instructions with fixed (32-bit) format and most register-based instructions.
Addressing modes	12–24.	Limited to 3–5.
General-purpose registers and cache design	8–24 GPRs, originally with a unified cache for instructions and data, recent designs also use split caches.	Large numbers (32–192) of GPRs with mostly split data cache and instruction cache.
CPI	CPI between 2 and 15.	One cycle for almost all instructions and an average CPI < 1.5.
CPU Control	Earlier microcoded using control memory (ROM), but modern CISC also uses hardwired control.	Hardwired without control memory.

3B)

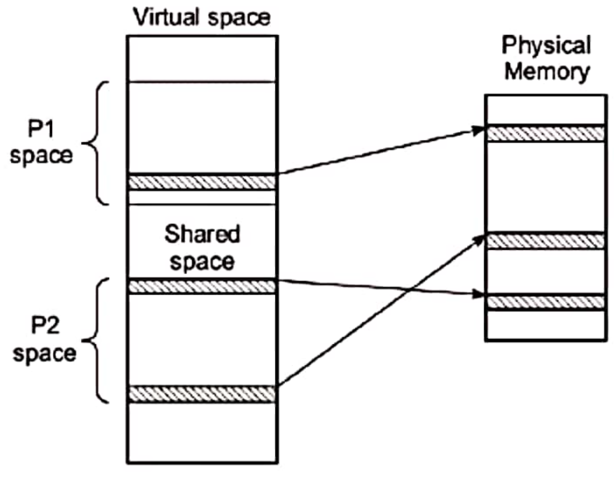
Virtual Memory Models

1. Private Virtual Memory

2. Shared Virtual Memory



(a) Private virtual memory space in different processors



(b) Globally shared virtual memory space

1. Private Virtual Memory

- In this scheme, each processor has a separate virtual address space, but all processors share the same physical address space.
- **Advantages:**
    - Small processor address space
    - Protection on a per-page or per-process basis
    - Private memory maps, which require no locking
- **Disadvantages**
    - The synonym problem – different virtual addresses in different/same virtual spaces point to the same physical page
    - The same virtual address in different virtual spaces may point to different pages in physical memory

2. Shared Virtual Memory

- All processors share a single shared virtual address space, with each processor being given a portion of it.
- Some of the virtual addresses can be shared by multiple processors.

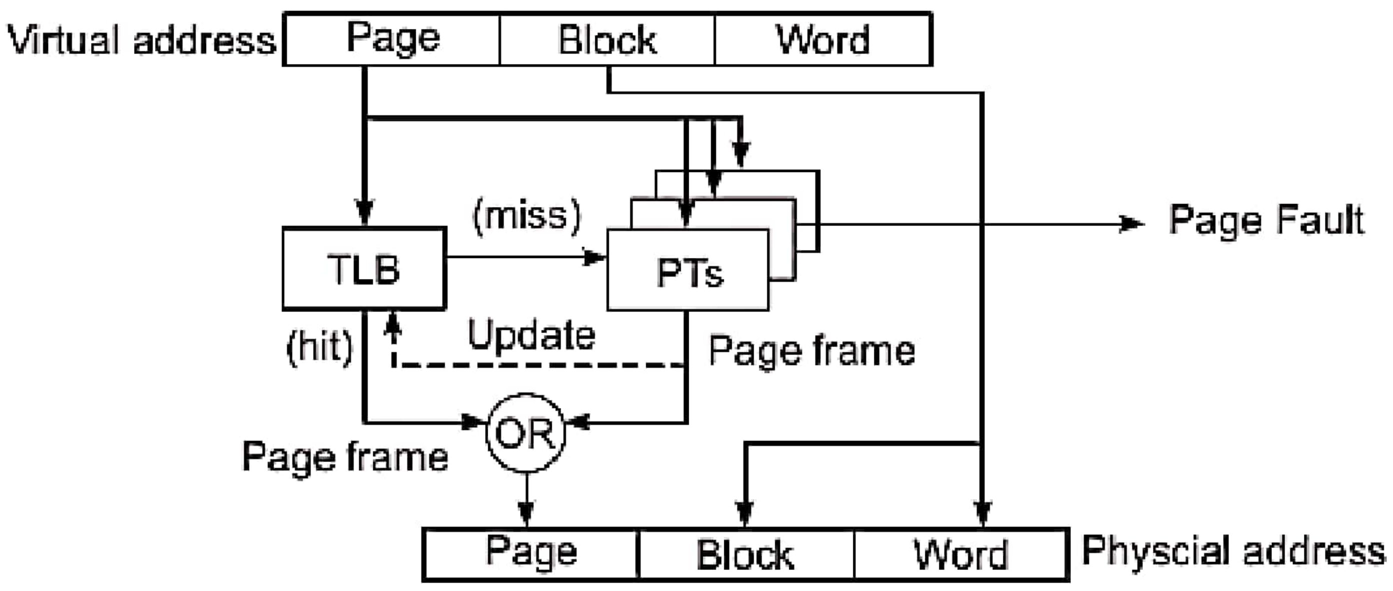
Advantages:

- All addresses are unique

3C)

Translation Lookaside Buffer

- The TLB is a high-speed lookup table which stores the most recently or likely referenced page entries.
- A *page entry* consists of essentially a (virtual page number, page frame number) pair. It is hoped that pages belonging to the same working set will be directly translated using the TLB entries.
- The use of a TLB and PTs for address translation is shown in Fig 4.21b. Each virtual address is divided into 3 fields:
    - The leftmost field holds the virtual page number,
    - the middle field identifies the cache block number,
    - the rightmost field is the word address within the block.



(b) Use of a TLB and PTs for address translation

- Our purpose is to produce the physical address consisting of the page frame number, the block number, and the word address.
- The first step of the translation is to use the virtual page number as a key to search through the TLB for a match.
- The TLB can be implemented with a special associative memory (content addressable memory) or use part of the cache memory.

4A)

Representative Superscalar Processors

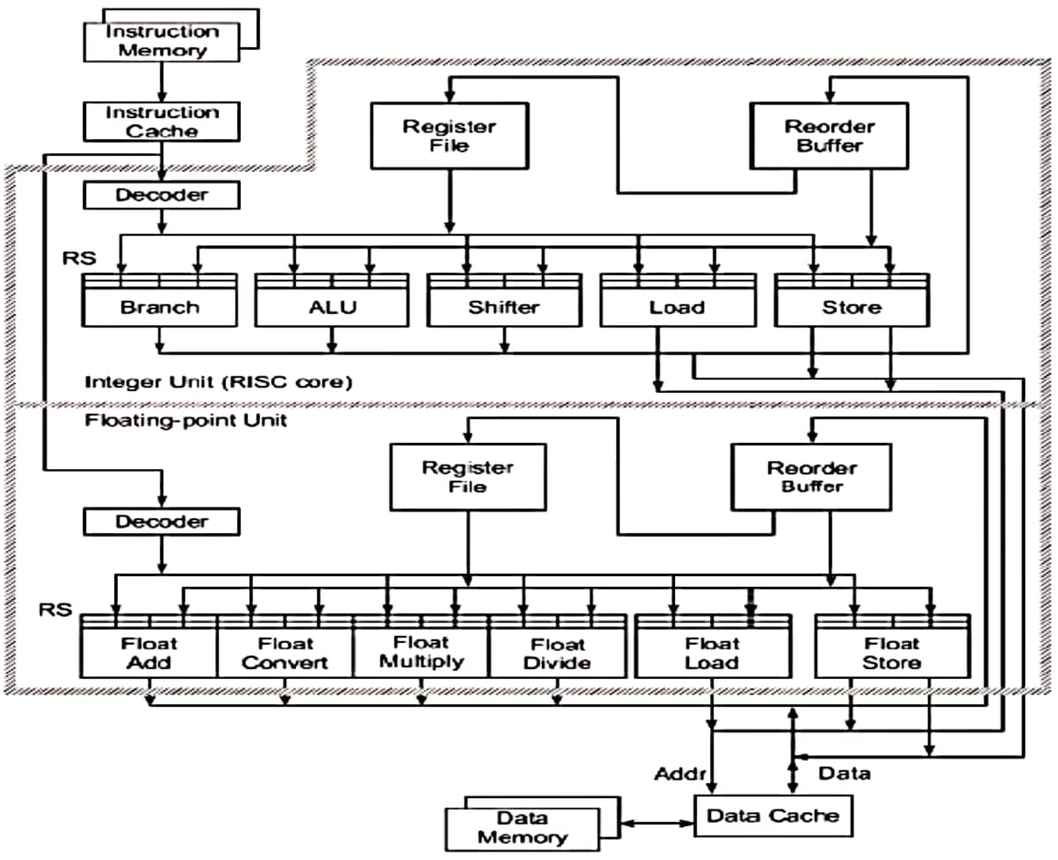


Fig. 4.12 A typical superscalar RISC processor architecture consisting of an integer unit and a floating-point unit (Courtesy of M. Johnson, 1991; reprinted with permission from Prentice-Hall, Inc.)

Typical Superscalar Architecture

- A typical superscalar will have
  - multiple instruction pipelines
  - an instruction cache that can provide multiple instructions per fetch
  - multiple buses among the function units
- In theory, all functional units can be simultaneously active.

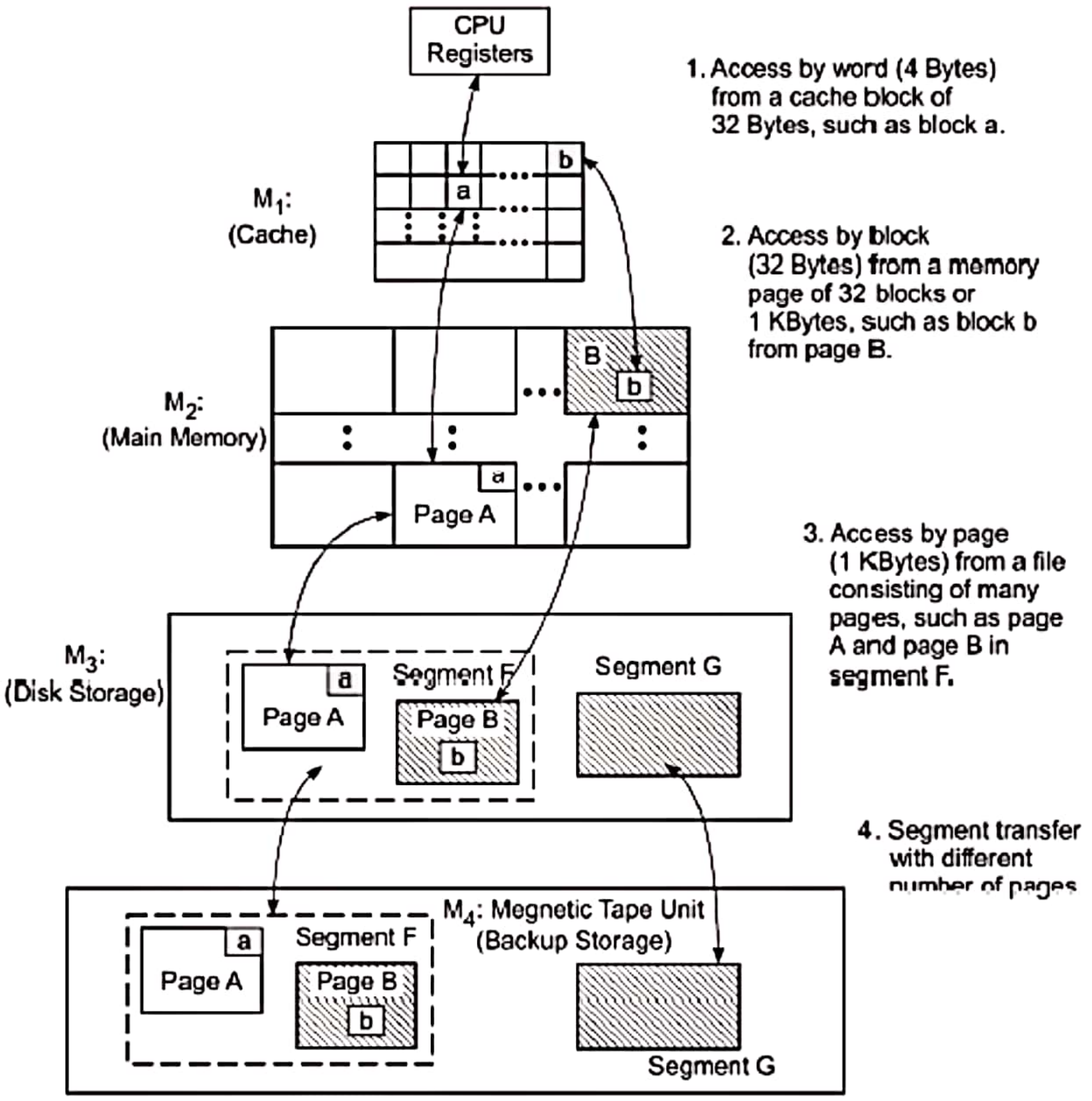
4B)

4.3.2 Inclusion, Coherence, and Locality

Information stored in a memory hierarchy (M1, M2,…, Mn) satisfies 3 important properties:

1. **Inclusion**
2. **Coherence**
3. **Locality**

- Program and data locality is characterized below as the foundation for using a memory hierarchy effectively.


CPU Registers
1. Access by word (4 Bytes) from a cache block of 32 Bytes, such as block a.
M_1: (Cache)
a
b
2. Access by block (32 Bytes) from a memory page of 32 blocks or 1 KBytes, such as block b from page B.
M_2: (Main Memory)
B
b
a
Page A
3. Access by page (1 KBytes) from a file consisting of many pages, such as page A and page B in segment F.
M_3: (Disk Storage)
Page A
a
Segment F
Page B
b
Segment G
4. Segment transfer with different number of pages
M_4: Megnetic Tape Unit (Backup Storage)
Page A
a
Segment F
Page B
b
Segment G

1. The Inclusion Property

- The inclusion property is stated as:

  $M_1 \subset M_2 \subset \ldots \subset M_n$
- The implication of the inclusion property is that all items of information in the "innermost" memory level (cache) also appear in the outer memory levels.

- The inverse, however, is not necessarily true. That is, the presence of a data item in level M_{i+1} does not imply its presence in level M_i. We call a reference to a missing item a "miss."

---

2. The Coherence Property

The requirement that copies of data items at successive memory levels be consistent is called the "coherence property."

Coherence Strategies

- **Write-through**
  - As soon as a data item in M_i is modified, immediate update of the corresponding data item(s) in M_{i+1}, M_{i+2}, … M_n is required.
  - This is the most aggressive (and expensive) strategy.
- **Write-back**
  - The update of the data item in M_{i+1} corresponding to a modified item in M_i is not updated unit it (or the block/page/etc. in M_i that contains it) is replaced or removed.
  - This is the most efficient approach, but cannot be used (without modification) when multiple processors share M_{i+1}, …, M_n.

3. Locality of References

- Memory references are generated by the CPU for either instruction or data access.
- These accesses tend to be clustered in certain regions in time, space, and ordering.

  There are three dimensions of the locality property:
  - *Temporal locality* – if location M is referenced at time t, then it (location M) will be referenced again at some time $t+\Delta t$.
  - *Spatial locality* – if location M is referenced at time t, then another location $M\pm\Delta m$ will be referenced at time $t+\Delta t$.
  - *Sequential locality* – if location **M** is referenced at time t, then locations **M+1, M+2, …** will be referenced at time $t+\Delta t$, $t+\Delta t'$, etc.

Module-2

3 a. Explain the architecture of VLIW processor and its pipeline operations. **(08 Marks)**

b. Explain the inclusion property and locality of reference along with its types in multilevel memory hierarchy. **(08 Marks)**

OR

4 a. Explain page replacement policies with the help of an example. **(08 Marks)**

b. Give the characteristics of symbolic processors. **(08 Marks)**

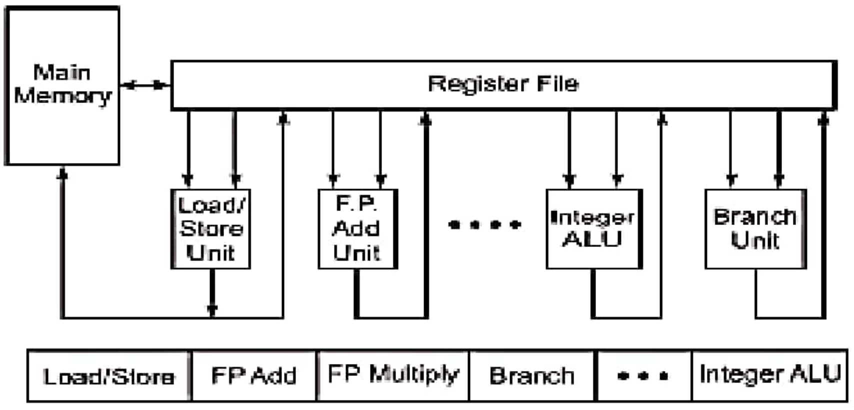
3A)

4.2.2 VLIW Architecture

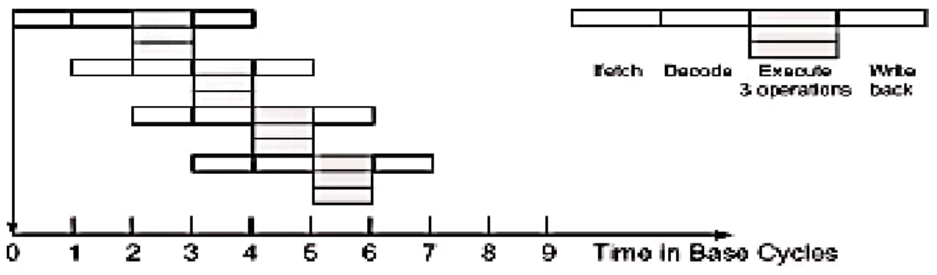
- VLIW = Very Long Instruction Word
- Instructions usually hundreds of bits long.
- Each instruction word essentially carries multiple "short instructions."
- Each of the "short instructions" are effectively issued at the same time.
- (This is related to the long words frequently used in microcode.)
- Compilers for VLIW architectures should optimally try to predict branch outcomes to properly group instructions.

Pipelining in VLIW Processors

- Decoding of instructions is easier in VLIW than in superscalars, because each "region" of an instruction word is usually limited as to the type of instruction it can contain.
- Code density in VLIW is less than in superscalars, because if a "region" of a VLIW word isn't needed in a particular instruction, it must still exist (to be filled with a "no op").
- Superscalars can be compatible with scalar processors; this is difficult with VLIW parallel and non-parallel architectures.



(a) A typical VLIW processor with degree $m = 3$



(b) VLIW execution with degree $m = 3$

Fig. 4.14 The architecture of a very long instruction word (VLIW) processor and its pipeline operations (Courtesy of Multiflow Computer, Inc., 1987)

4A)

4.4.3 Page Replacement Policies

- Memory management policies include the allocation and deallocation of memory pages to active processes and the replacement of memory pages.
- Demand paging memory systems. refers to the process in which a resident page in main memory is replaced by a new page transferred from the disk.
- Since the number of available page frames is much smaller than the number of pages, the frames will eventually be fully occupied.

The following page replacement policies are specified in a demand paging memory system for a page fault at time *t*.

(1) **Least recently used (LRU)**—This policy replaces the page in R(t) which has the longest backward distance:

$$q(t)=y, \quad \text{iff} \quad b_t(y)=\max_{x\in R(t)}\{b_t(x)\}$$

(2) **Optimal (OPT) algorithm**—This policy replaces the page in R(t) with the longest forward distance:

$$q(t)=y, \quad \text{iff} \quad f_t(y)=\max_{x\in R(t)}\{f_t(x)\}$$

(3) **First-in-first-out (FIFO)**—This policy replaces the page in R(t) which has been in memory for the longest time.

(4) **Least frequently used (LFU)**—This policy replaces the page in R(t) which has been least referenced in the past.

(5) **Circular FIFO**—This policy joins all the page frame entries into a circular FIFO queue using a pointer to indicate the front of the queue.

- An allocation bit is associated with each page frame. This bit is set upon initial allocation of a page to the frame.

(6) **Random replacement**—This is a trivial algorithm which chooses any page for replacement randomly.

Example:

Consider a paged virtual memory system with a two-level hierarchy: main memory M_1 and disk memory M_2.

Assume a page size of four words. The number of page frames in M_1 is 3, labeled a, b and c; and the number of pages in M_2 is 10, identified by 0, 1, 2,….9. The ith page in M_2consists of word addresses 4i to 4i + 3 for all i = 0, 1, 2, …, 9.

A certain program generates the following sequence of word addresses which are grouped (underlined) together if they belong to the same page. The sequence of page numbers so formed is the *page trace*:

Word trace:	0,1,2,3,	4,5,6,7,	8,	16,17,	9,10,11,	12,	28,29,30,	8,9,10,	4,5,	12,	4,5
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
Page trace:	0	1	2	4	2	3	7	2	1	3	1

Page tracing experiments are described below for three page replacement policies: LRU, OPT, and FIFO, respectively. The successive pages loaded in the page frames (PFs) form the trace entries. Initially, all PFs are empty.

	PF	0	1	2	4	2	3	7	2	1	3	1	Hit Ratio
LRU	*a*	0	0	0	4	4	4	7	7	7	3	3	$\frac{3}{11}$
	b		1	1	1	1	3	3	3	1	1	1	
	c			2	2	2	2	2	2	2	2	2	
	Faults	*	*	*	*		*	*		*	*		
OPT	*a*	0	0	0	4	4	3	7	7	7	3	3	$\frac{4}{11}$
	b		1	1	1	1	1	1	1	1	1	1	
	c			2	2	2	2	2	2	2	2	2	
	Fault	*	*	*	*		*	*			*		
FIFO	*a*	0	0	0	4	4	4	4	2	2	2	2	$\frac{2}{11}$
	b		1	1	1	1	3	3	1	1	1	1	
	c			2	2	2	2	7	7	7	3	3	
	Faults	*	*	*	*		*	*	*	*	*		

4B)

Symbolic Processors

- Symbolic processors are somewhat unique in that their architectures are tailored toward the execution of programs in languages similar to LISP, Scheme, and Prolog.
- In effect, the hardware provides a facility for the manipulation of the relevant data objects with "tailored" instructions.
- These processors (and programs of these types) may invalidate assumptions made about more traditional scientific and business computations.

Table 4.6 *Characteristics of Symbolic Processing*

Attributes	*Characteristics*
Knowledge Representations	Lists, relational databases, scripts, semantic nets, frames, blackboards, objects, production systems.
Common Operations	Search, sort, pattern matching, filtering, contexts, partitions, transitive closures, unification, text retrieval, set operations, reasoning.
Memory Requirements	Large memory with intensive access pattern. Addressing is often content -based. Locality of reference may not hold.
Communication Patterns	Message traffic varies in size and destination; granularity and format of message units change with applications.
Properties of Algorithms	Nondeterministic, possibly parallel and distributed computations. Data dependences may be global and irregular in pattern and granularity.
Input/Output requirements	User-guided programs; intelligent person-machine interfaces; inputs can be graphical and audio as well as from keyboard; access to very large on-line databases.
Architecture Features	Parallel update of large knowledge bases, dynamic load balancing; dynamic memory allocation; hardware-supported garbage collection; stack processor architecture; symbolic processors.

Module-2

3 a. Distinguish between typical RISC and CISC process architectures. (08 Marks)
b. With a diagrams, explain the models of a basic scalar computer system. (08 Marks)

OR

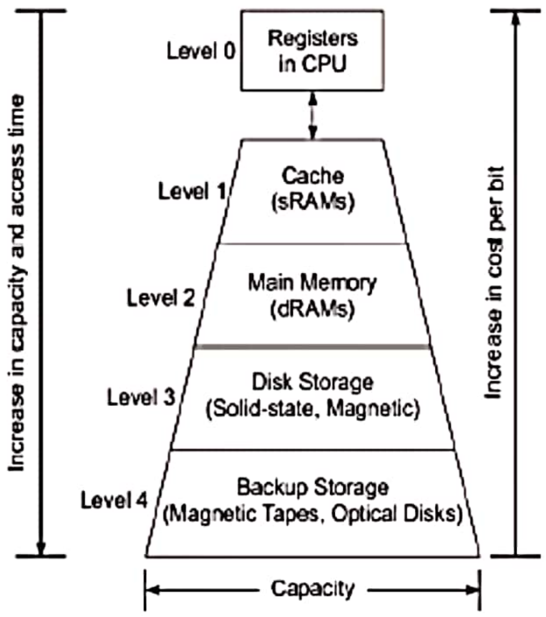
4 a. With a diagram, explain a typical superscalar RISC processor architecture consisting of an integer unit and a floating point unit. (10 Marks)
b. With a diagram, explain the hierarchical memory technology. (06 Marks)

4B)

4.3 Memory Hierarchical Technology

- Storage devices such as registers, caches, main memory, disk devices, and backup storage are often organized as a hierarchy as depicted in Fig. 4.17.
- The memory technology and storage organization at each level is characterized by five parameters:

1. **access time t_i** (round-trip time from CPU to ith level)
2. **memory size s_i** (number of bytes or words in level i)
3. **cost per byte c_i**
4. **transfer bandwidth b_i** (rate of transfer between levels)
5. **unit of transfer x_i** (grain size for transfers between levels i and i+1)



Increase in capacity and access time
Level 0
Registers in CPU
Level 1
Cache (sRAMs)
Level 2
Main Memory (dRAMs)
Level 3
Disk Storage (Solid-state, Magnetic)
Level 4
Backup Storage (Magnetic Tapes, Optical Disks)
Capacity
Increase in cost per bit

Fig. 4.17 A four-level memory hierarchy with increasing capacity and decreasing speed and cost from low to high levels

Memory devices at a lower level are:

- faster to access,
- are smaller in capacity,
- are more expensive per byte,
- have a higher bandwidth, and
- have a smaller unit of transfer.

In general, $t_{i-1} < t_i$, $s_{i-1} < s_i$, $c_{i-1} > c_i$, $b_{i-1} > b_i$ and $x_{i-1} < x_i$ for i = 1, 2, 3, and 4 in the hierarchy where i = 0 corresponds to the CPU register level.

Caches

- The cache is controlled by the MMU and is programmer-transparent.
- The cache can also be implemented at one or multiple levels, depending on the speed and application requirements.
- Multi-level caches are built either on the processor chip or on the processor board.
- Multi-level cache systems have become essential to deal with memory access latency.

Main Memory (Primary Memory)

- It is usually much larger than the cache and often implemented by the most cost-effective RAM chips, such as DDR SDRAMs, i.e. dual data rate synchronous dynamic RAMs.
- The main memory is managed by a MMU in cooperation with the operating system.

Disk Drives and Backup Storage

- The disk storage is considered the highest level of on-line memory.
- It holds the system programs such as the OS and compilers, and user programs and their data sets.
- Optical disks and magnetic tape units are off-line memory for use as archival and backup storage.
- They hold copies of present and past user programs and processed results and files.
- Disk drives are also available in the form of RAID arrays.

Peripheral Technology

- Peripheral devices include printers, plotters, terminals, monitors, graphics displays, optical scanners, image digitizers, output microfilm devices etc.
- Some I/O devices are tied to special-purpose or multimedia applications.