

MODULE 2

PROCESSORS & MEMORY HIERARCHY

CONTENTS

Processors and memory hierarchy

- **Advanced processor technology**
 - ▣ **Design Space of processors**
 - ▣ **Instruction -set Architecture**
 - ▣ **CISC Scalar Processors**
 - ▣ **RISC Scalar Processors**
- **Super scalar and vector processors**
 - ▣ **Super Scalar Processors**
 - ▣ **Vector and Symbolic Processors**
- **Memory hierarchy technology.**

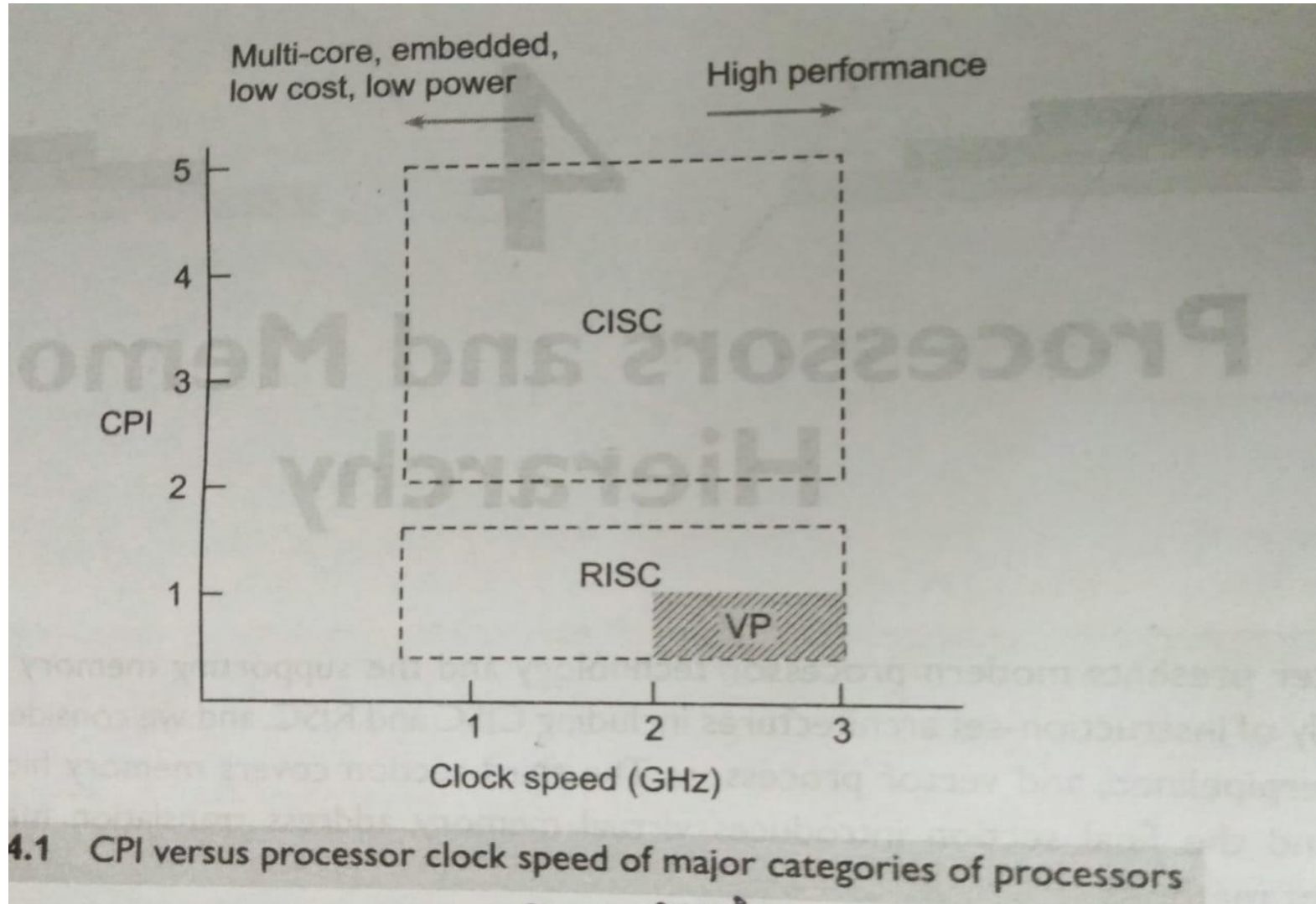


ADVANCED PROCESSOR TECHNOLOGY

- Major processor families are:-
 - CISC
 - RISC
 - Superscalar
 - VLIW
 - Super pipelined
 - Vector
 - Symbolic processors
- **Scalar and vector processors** are used for numerical computations
- **Symbolic processors** are used for AI applications



Design space of processors



DESIGN SPACE OF PROCESSORS

- **CPI** Vs **clock speed** of various processors
 - Processors designed for following aims **have lower clock speeds**
 - **Multi-core chips**
 - **Embedded applications**
 - Low cost applications
 - Low power consumption
 - High performance processors are designed to operate at **higher clock speeds**



DESIGN SPACE

- **CISC architecture** design space
 - **Clock rate** of CISC processors ranges upto few GHz
 - **CPI** of CISC instructions varies from 1 to 20
 - Eg: Intel Pentium, M68040
- **RISC architecture design space**
 - CPI of RISC instruction is reduced between 1 and 2 cycles
 - Eg:
 - **SPARC** (Scalable Processor Architecture)
 - **MIPS** (Microprocessor without Interlocked Pipeline Stages)



○ **Superscalar processor** design space

- Subclass of RISC processor
- Multiple instructions are issued simultaneously during each cycle
- Effective **CPI is lower** than scalar RISC
- Clock rate matches with that of scalar RISC

○ **Design space of VLIW architecture**

- Use more functional units than superscalar processor
- **CPI** is further lowered
- Eg: Intel i860



○ **Vector supercomputers** design space

- Use multiple functional units
- Perform concurrent scalar and vector operations
- **CPI** is very low



INSTRUCTION PIPELINES [1]

- **Execution cycle** of an instruction involves 4 phases
 - ▣ **Fetch**
 - ▣ Decode
 - ▣ Execute
 - ▣ Write back
- **Pipeline** receives successive instructions from one end
- It executes them in a streamlined overlapped fashion



INSTRUCTION PIPELINES [2]

○ Pipeline cycle

- It is defined as the **time required for each phase to complete its operation** assuming equal delay in all phases

○ Instruction pipeline cycle

- It is the clock period of the instruction pipeline

○ Instruction issue latency

- Time required between the issue of **two adjacent instructions**

○ Instruction issue rate

- **No: of instructions** issued per cycle



INSTRUCTION PIPELINES [3]

○ Simple operation latency

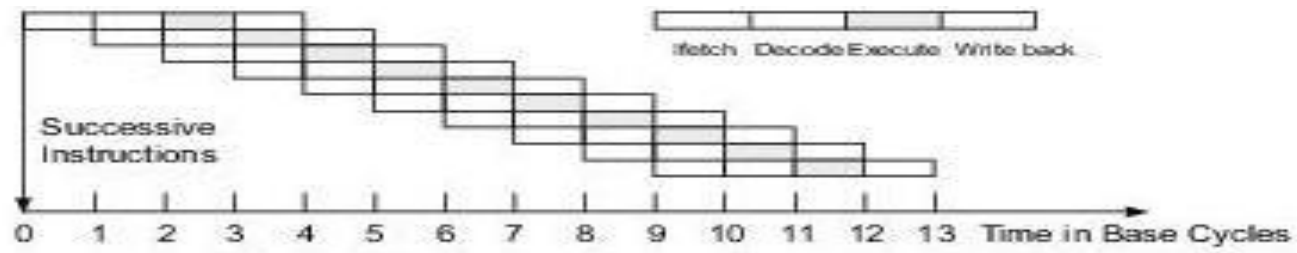
- No: of cycles needed to perform simple operations
- Simple operations are:-
 - Integer adds
 - Loads
 - Stores
 - **Branches**
 - **moves**

○ Resource conflicts

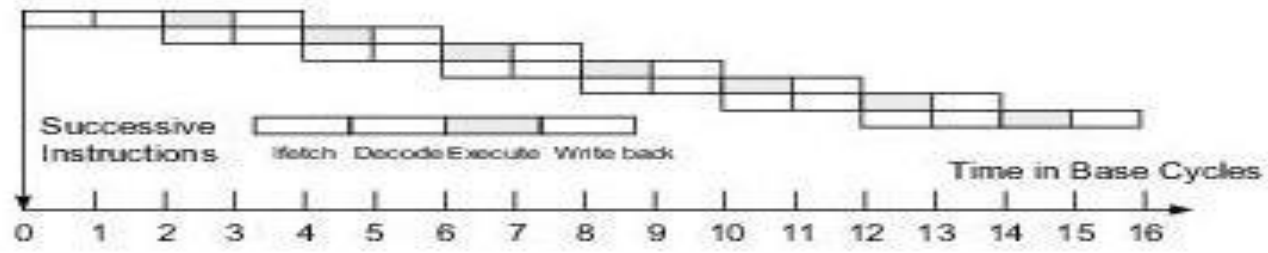
- It is a situation where 2 or more instructions demand use of the same functional unit at the **same time**



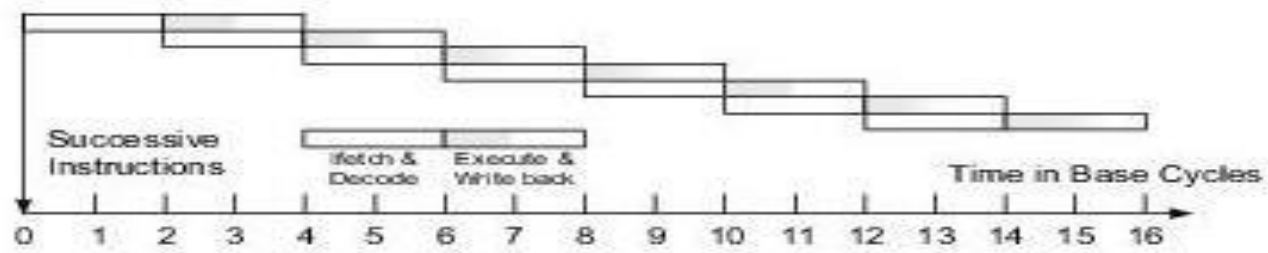
INSTRUCTION PIPELINES [4]



(a) Execution in a base scalar processor



(b) Underpipelined with two cycles per instruction issue



(c) Underpipelined with twice the base cycle



BASIC SCALAR PROCESSOR

- It is a machine with following features:-
 - 1 instruction issued per cycle
 - 1 cycle latency for simple operation
 - 1 cycle latency between instruction issues



INSTRUCTION PIPELINES [5]

- Another **under-pipelined** situation is in which the pipeline cycle time is doubled by combining pipeline stages.
- In this case, the fetch and decode phases are combined into one pipeline stage, and execute and write back are combined into another stage.
- This will also result in **poor pipeline** utilization.
- The effective CPI rating is 1 for the ideal pipeline
- The effective CPI rating is 2 for fig b
- The effective CPI rating is one half for fig c



INSTRUCTION SET ARCHITECTURES

- Instruction set of a computer specifies
 - **Machine instructions** that a programmer can use while programming the machine
- **Complexity** of an instruction depends on:-
 - Instruction format
 - Data format
 - Addressing modes
 - General purpose registers
 - Opcode specifications
 - Flow control mechanisms



TYPES OF INSTRUCTION SET ARCHITECTURES

- 2 types
 - **Complex** instruction set
 - **Reduced** instruction set



COMPLEX INSTRUCTION SETS

- CISC contains **120 to 350** instructions
- Instructions use **variable instruction** and **data formats**
- Use small set of 8 to 24 general purpose registers
GPR
- Executes large no: of **memory reference** operations
- More than a dozen **addressing modes**



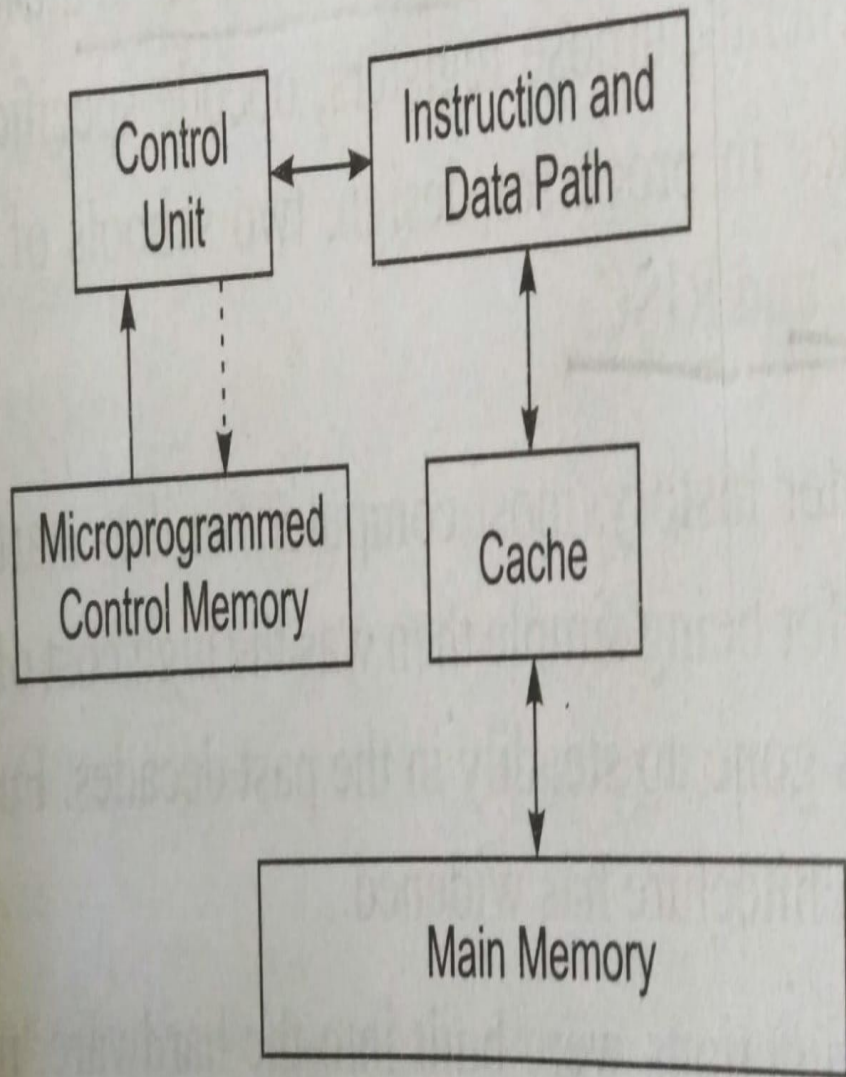
REDUCED INSTRUCTION SET

- Contains less than **100** instructions
- Have **fixed instruction formats** (32bits)
- Only **3 to 5 simple** addressing modes are used
- Most instructions are **register** based
- Memory access is done by load/store instructions only
- Most instructions execute in **one cycle**
- **Hardwired control** is used
- Higher **clock rate**
- Lower **CPI**
- Higher processor performance

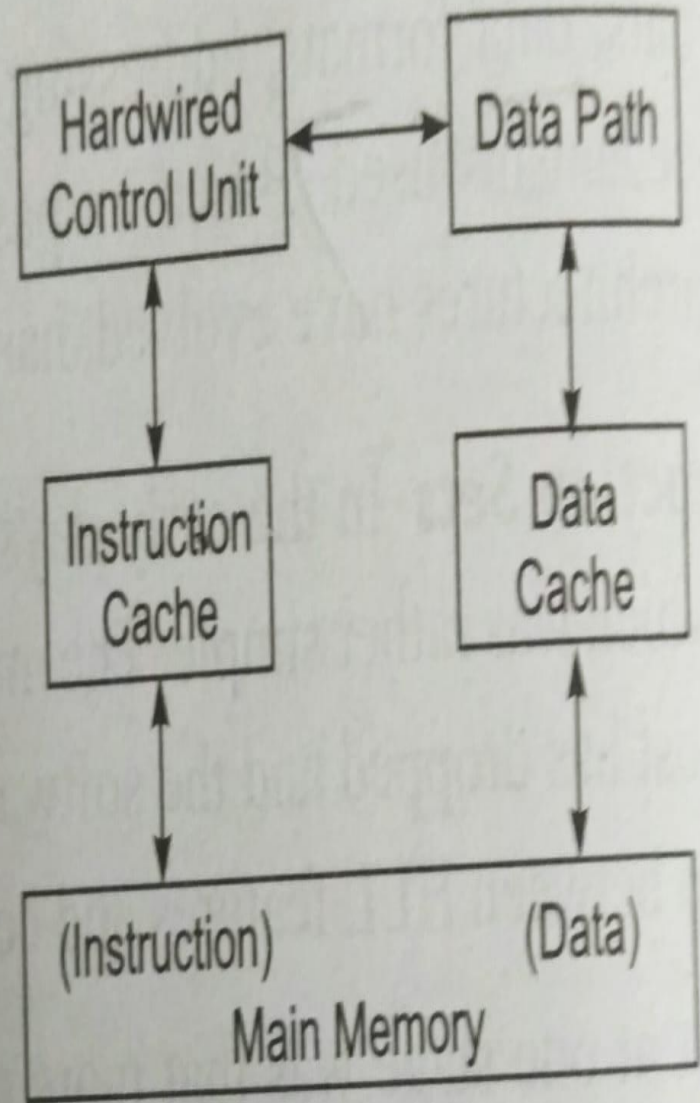


ARCHITECTURAL DISTINCTIONS

CISC architecture	RISC architecture
Uses unified cache for holding both instructions and data	Uses separate instruction cache and data cache
same data/instruction	different
Use of micro-programmed control	Use hardwired control
Control memory was needed in early CISC	No need of control memory
High CPI	Less CPI



(a) The CISC architecture with microprogrammed control and unified cache



(b) The RISC architecture with hardwired control and split instruction cache and data cache



CISC SCALAR PROCESSOR

INTRODUCTION

- **Scalar processor** executes scalar data
- **Simplest scalar processor**
 - executes integer instruction using fixedpoint operands
- **Capable scalar processor**
 - Execute both integer & floating point operations
- **Modern scalar processor**
 - Posses both integer unit & floating point unit
- CISC uses pipelined design



REPRESENTATIVE **CISC**

- VAX 8600
- Motorola MC68040
- Intel i486



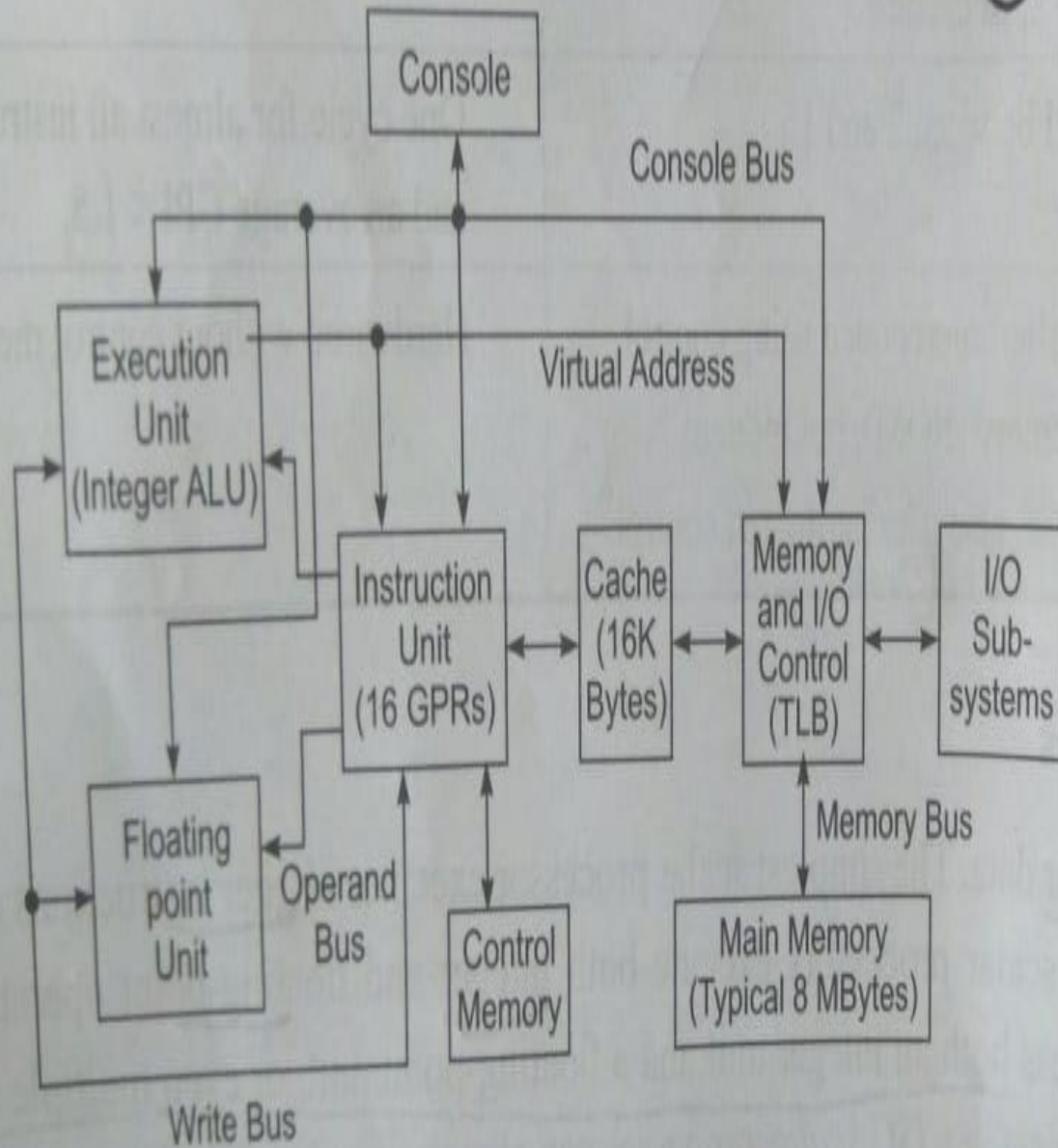
VAX 8600

- Introduced by **Digital Equipment Corporation**
- Uses **Micro programmed control**
- **300** instructions
- **20** addressing mode
- **16** GPRs
- CPI range from 2 to 20 cycles
- CPU has **2 functional** units
 - Execution unit
 - Floating point unit
- Used for **concurrent execution** of integer & floating point instructions
- **Unified cache** used for holding data and instruction



- Pipelining has 6 stages
- **Instruction unit**
 - It **pre-fetches & decoded** the instructions
 - Handles **branching** operations
 - Supply operands to **2 functional units** in a pipelined manner
- **Translation look aside buffer TLB**
 - Used in memory control unit for the fast generation of physical address from virtual address





Captions:

CPU = Central Processor Unit

TLB = Translation Lookaside Buffer

GPR = General Purpose Register

MOTOROLA MC68040

- Consist of **1.2** million **transistors**
- **100** instructions
- **16** GPR
- **18** addressing modes
- Separate instruction & data cache
 - 4Kb instruction cache
 - 4Kb data cache
- Separate **MMU** supported by **ATC**(address translation cache)
 - **This is equivalent to TLB**
- Data format range **from 8 to 80** bits



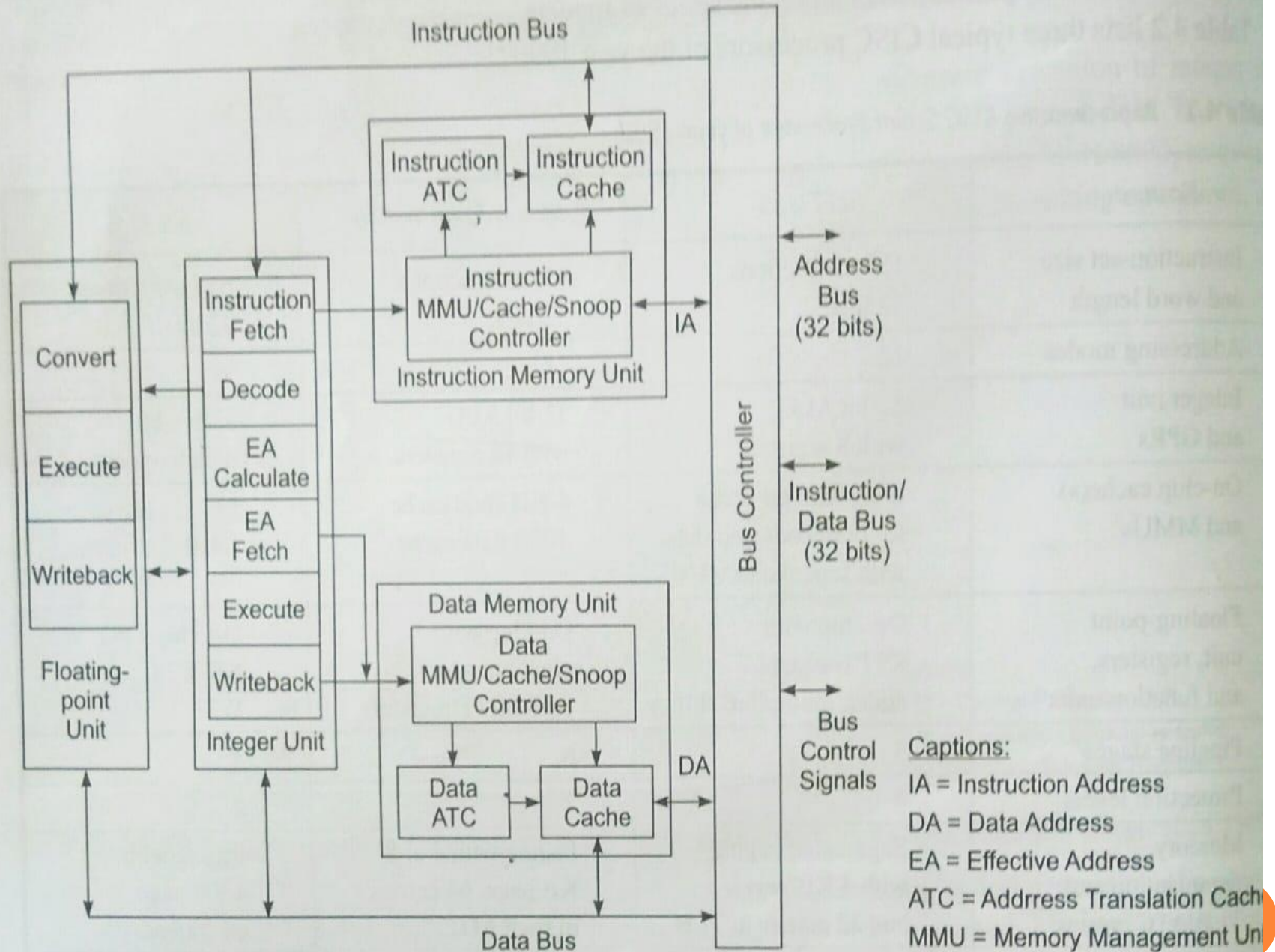


Fig. 4.6 Architecture of the MC68040 processor (Courtesy of Motorola Inc., 1991)

MOTOROLA M68040[2]

- CPU has 2 units
 - Integer unit
 - Floating point unit
- **Integer unit**
 - Integer unit has 6 stage instruction pipeline
 - All instructions are decoded in this unit
 - Floating point instructions are forwarded to floating point unit for execution
- **Floating point unit**
 - It consist of 3 pipeline stages



MOTOROLA M68040 [3]

- **Memory units**

- Data memory unit
- Instruction memory unit

- Separate **instruction & data buses** are used for instruction & data memory units
- Both buses are 32 bitwide

- Each of 2 **ATC** have 64 entries

- This provide **fast translation** from **virtual** to **physical addresses**



CISC MICROPROCESSOR FAMILIES

- Intel family
 - Intel 4004 → 4 bit
 - Intel 8008,8080,8085 → 8 bit
 - Intel 8086,80186,80268 → 16 bit
 - Intel 80386,80486 → 32 bit
- Motorola family
 - MC6800 → 8 bit
 - MC68000 → 16 bit
 - MC68020 → 32 bit



RISC SCALAR PROCESSORS

32

INTRODUCTION

- Generic RISC processors are called as **scalar RISC**
- They are designed to issue **one instruction per cycle**
- Similar to **Base scalar processor**
- A good compiler is required in RISC processor than in CISC

REPRESENTATIVE RISC PROCESSORS

- Sun SPARC
 - Intel i860
 - Motorola M88100
-
- All of these use 32 bit instructions
 - SPARC stands for Scalable Processor Architecture

SUN SPARC ARCHITECTURE

- These processors are implemented by a **no: of licensed manufacturers**
- All of them implemented floating point unit on a separate coprocessor chip
- Sun SPARC instruction set has 69 basic instructions
- It runs each procedure using 32 bit registers
 - 8 of these registers are **global registers** shared by all procedures
 - Remaining 24 are window registers associated with only each procedure

INTEL 860 PROCESSOR ARCHITECTURE

- 64 bit RISC processor fabricated on a single chip
- It executed **82 instructions**
- They included:-
 - **42 RISC integer**
 - **24 floating point**
 - **10 graphics**
 - **6 assembler pseudo operations**
- All instructions are executed in **one cycle**
- Applications
 - **Used in floating point accelerators**
 - **Graphics subsystem**
 - **Workstations**
 - **Multiprocessors**
 - **multicomputers**

ARCHITECTURE OF INTEL 860

- There are 9 functional units
 - Bus control unit
 - Data cache
 - Instruction cache
 - MMU
 - Integer unit
 - Floating point control unit
 - Pipelined adder unit
 - Pipelined multiplier unit
 - Graphics unit
- These units are interconnected using multiple data paths
 - Width ranges from 32 to 128 bits

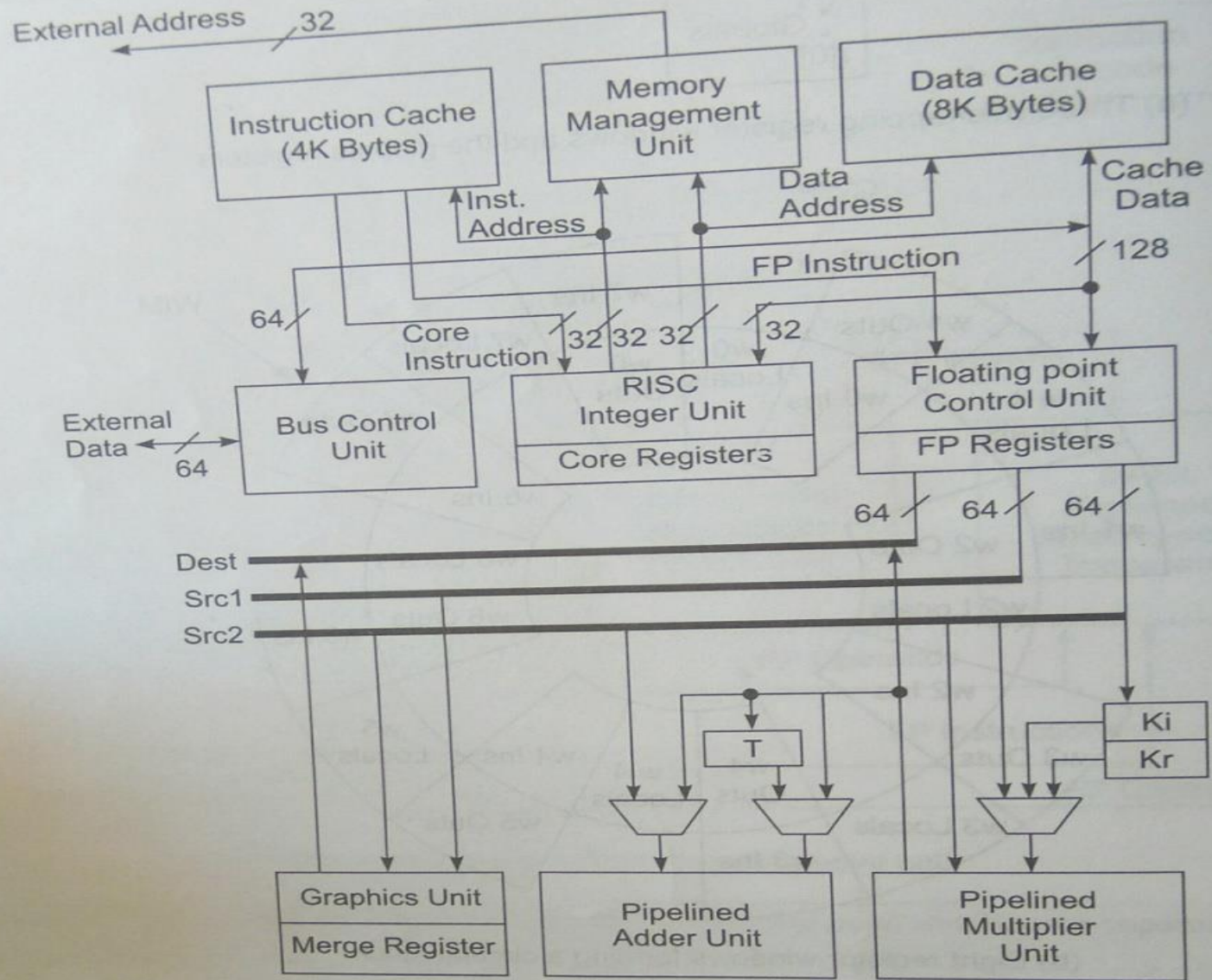


Fig. 4.9 Functional units and data paths of the Intel i860 RISC microprocessor (Courtesy of Intel Corporation 1990)

○ **Bus control unit**

- External and internal **address buses** are 32 bit wide
- External and internal **data buses** are 64 bits wide
- It coordinated 64 bit data transfer b/w chip and the outside world

○ **Instruction cache**

- It is 4Kb cache
- Organized as 2 way set associative memory
- Each cache block have a size of **32 bytes**
- It transferred 64 bits per **clock cycle**

○ Data cache

- It is 8 kb cache
- 2 way set associative
- It transferred 128 bits per clock cycle
- Write back policy was used

○ MMU

- This unit implemented 4Kb paged virtual memory via TLB

○ RISC integer unit

- RISC integer ALU is 32bits
- It executed
 - Load
 - Store
 - Integer
 - Bit
 - Control instructions
- It fetches instructions for floating point control unit also

○ Floating point units

- There are 2 floating point units
 - Multiplier unit
 - Adder unit
- They can be used separately or simultaneously
- These units are controlled by **floating point control unit**
- **Dual operation floating point** instructions use both adder & multiplier units in parallel
 - Eg: add-and-multiply instruction, Subtract- and- multiply instruction

- Both integer unit & floating point control unit execute concurrently
- Floating point unit operates with 32 bit & 64 bit operands

- **Graphics unit**
 - This unit executed integer operations corresponding to **8, 16 or 32 bit pixel data types**
 - Supported 3D drawing in graphics frame buffer
 - It also supported:-
 - **Color intensity**
 - **Shading**
 - **Hidden surface elimination**

- **Merge register**
 - Used only by vector integers instructions
 - This register accumulated results of **multiple addition operations**



SUPERSCALAR PROCESSORS

INTRODUCTION

- A CISC or a RISC scalar processor can be improved with a **superscalar or vector architecture**.
- **Scalar processors** are those executing **one instruction per cycle**.
- Only one instruction is issued per cycle, and only one completion of instruction is expected from the pipeline per cycle.
- In a **superscalar processor**, **multiple instructions are issued** per cycle and **multiple results** are generated per cycle



PIPELINING IN SUPERSCALAR PROCESSOR

- Superscalar processors were originally developed as an alternative to vector processors
- They exploit **higher degree** of instruction level parallelism.
- A **superscalar processor of degree m** can issue **m instructions** per cycle.
- The **base scalar processor**, implemented either in RISC or CISC, has $m = 1$.
- Instruction issue **degree** is limited to **2 to 5**
- Thus m instructions must be executable in parallel.



PIPELINING IN SUPERSCALAR PROCESSOR

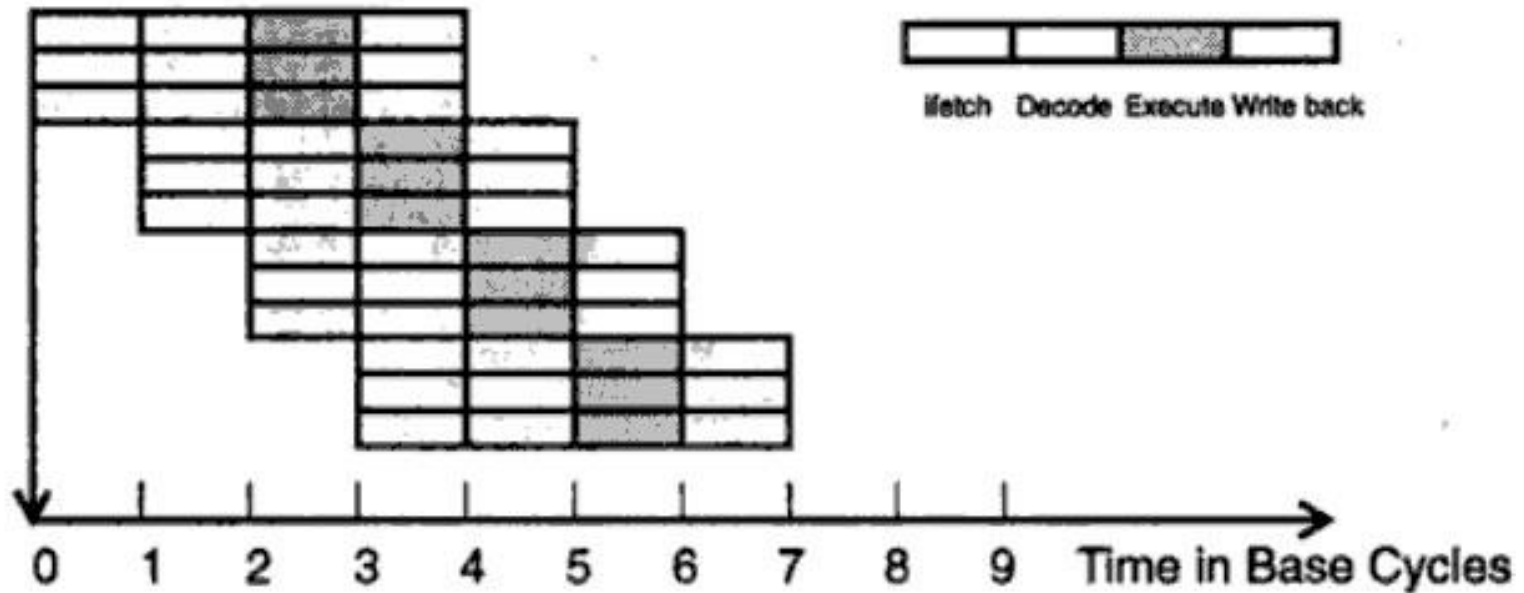


Fig: A superscalar processor of degree $m=3$



PIPELINING IN SUPERSCALAR PROCESSOR

- In a **superscalar processor**; the simple operation latency should require only one cycle, as in the **base scalar processor**.
- Due to the desire for a higher degree of instruction-level parallelism in programs, the superscalar processor depends more on an optimizing compiler to exploit parallelism



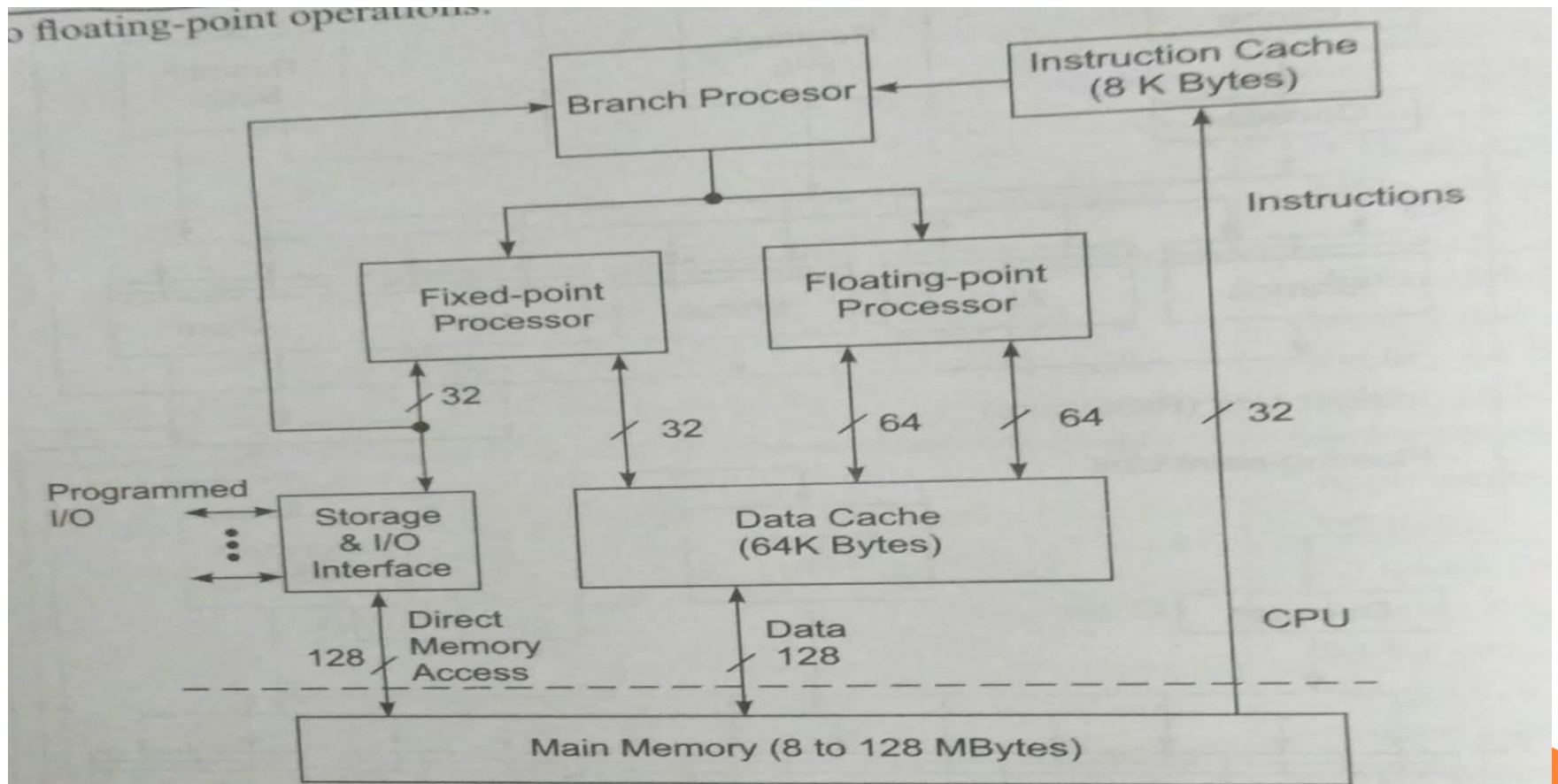
REPRESENTATIVE SUPERSCALAR PROCESSOR R

- IBM RS/6000
- DEC alpha 21064
- Intel i960CA

IBM RS/6000 ARCHITECTURE

- It has 3 functional units
 - **Branch processor**
 - **Fixed point unit FXU**
 - **Floating point unit FPU**
- These could operate in parallel
- **Branch processor**
 - This unit could arrange the execution of up to 5 instructions per cycle
 - The instructions included are:-
 - 1 **branch instruction** → executed by branch processor
 - 1 **fixed point instruction** → done by FXU
 - 1 **condition register instruction** → done by branch processor
 - 1 **floating point multiply-add instruction** → done by FPU
 - This inst can be counted as 2 floating point operations

IBM RS/6000 ARCHITECTURE



The POWER architecture of the IBM RISC System/6000 superscalar processor (International Business Machines Corporation, 1990)

- RS/6000 used **hardwired control logic**
- System used no: of buses
 - ▣ **32 bit for FXU**
 - ▣ **64 bit for FPU**
 - ▣ **128 bit for I cache and D cache**
- Application
 - ▣ Numerically intensive scientific & engineering applications
 - ▣ Multiuser commercial environments



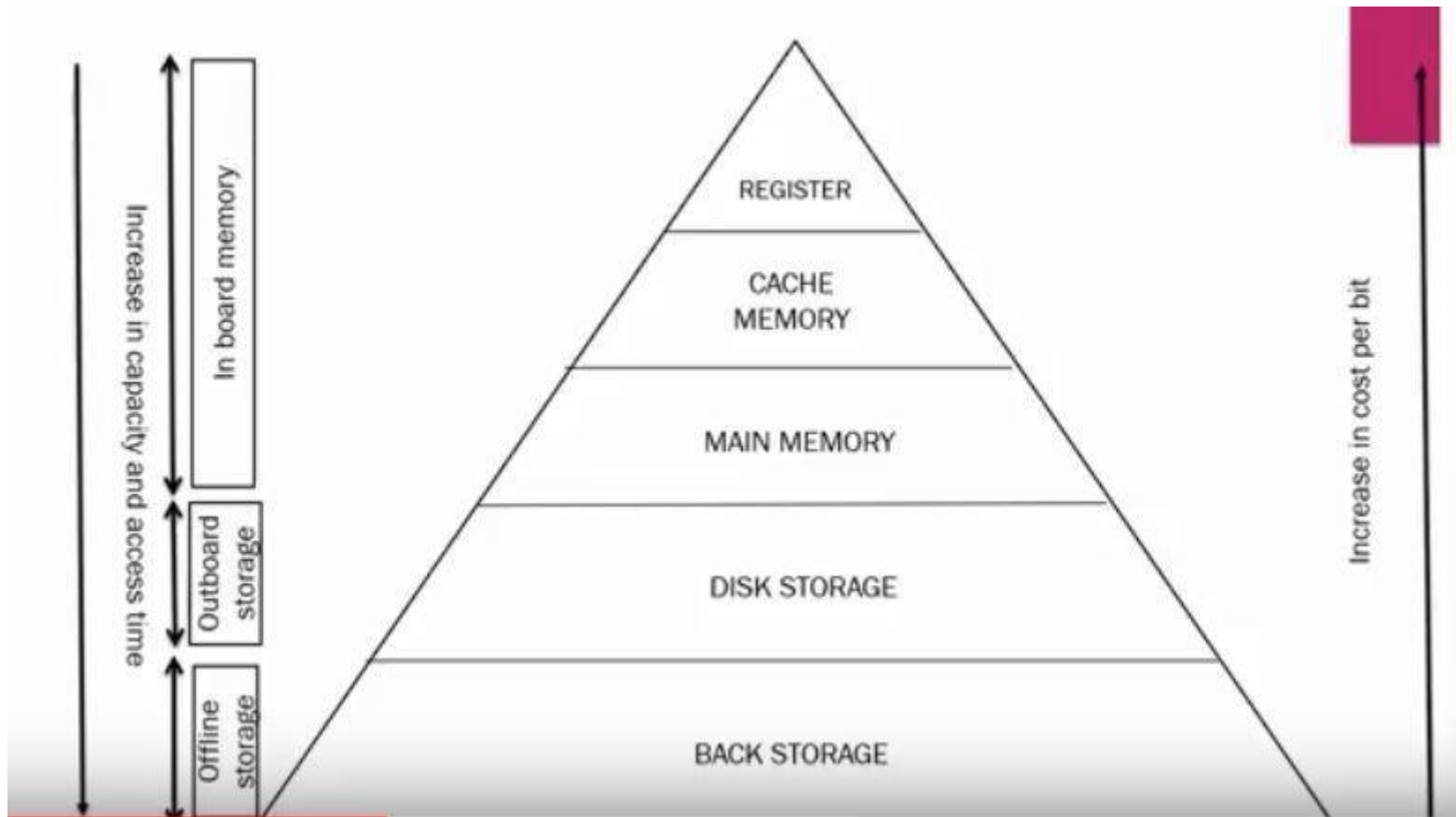
MEMORY HIERARCHY TECHNOLOGY

HIERARCHICAL MEMORY TECHNOLOGY

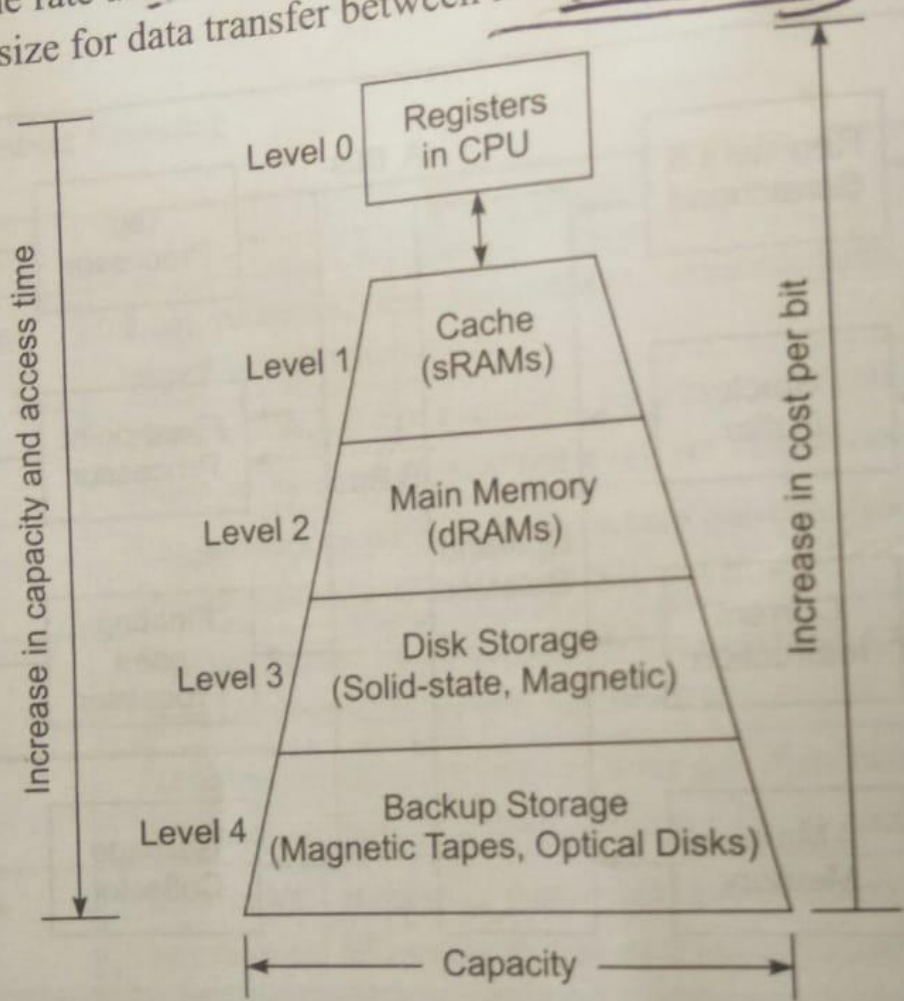
- In computer architecture the **memory hierarchy** separates computer storage into hierarchy based on **response time**
- Since **Response time , complexity and capacity** are related the level may also be distinguished by their performance and controlling technology
- **Storage devices** are organized into a hierarchy of layers
 - **Registers**
 - **Caches**
 - **Main memory**
 - **Disk devices**
 - **Backup storage**



MEMORY HIERARCHY



width b_i refers to the rate
refers to the grain size for data transfer between



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

PARAMETERS OF MEMORY TECHNOLOGIES

- Memory technology at each level is characterized by 5 parameters
- **Access time (t_i)**
 - It refers to the round-trip time from CPU to the i^{th} level memory
- **Memory size (s_i)**
 - No: of bytes or words in level i
- **Cost per byte (c_i)**
 - Cost of i^{th} level memory
 - It is estimated by the product of $c_i \cdot s_i$
- **Bandwidth (b_i)**
 - It refers to the **rate** at which information is transferred between adjacent levels
- **Unit of transfer (x_i)**
 - Refers to the grain **size** for data transfer between levels i and level $i+1$

ANALYSIS OF PARAMETERS

- Memory devices at a **lower level** are :-
 - faster to access
 - Smaller in size
 - More expensive per byte
 - Higher bandwidth
 - Smaller units of transfer as compared with higher levels
- $t_{i-1} < t_i, s_{i-1} < s_i, c_{i-1} > c_i, b_{i-1} > b_i, x_{i-1} < x_i$



MEMORY HIERARCHY

○ Registers and caches

- Registers are part of the **processors**
- Compiler does the register assignment
- **Register transfer operations** are directly controlled by the processor
- Transfer is conducted at processor speed in one clock cycle

○ Caches

- Multilevel caches are built either on processor chip or on the processor board
- They are controlled by **MMU** & are programmer transparent
- They are implemented in 1 or **multiple levels**
- Speed of processor is faster than memory speed



○ **Main memory**

- Also called **primary memory**
- Larger than the cache
- Implemented by cost effective **RAM** chips like DDR SDRAM
- Managed by **MMU** with OS

○ **Disk drives**

- Disk storage is the highest level of on-line memory
- Holds system programs like **OS, compilers, user programs and data sets**



○ **Optical disk & magnetic tape** (backup storage)

- They are **offline memory**
- Used for archival and backup storage
- They hold copies of present & past user programs, processed results & files



Table 4.7 Memory Characteristics of a Typical Mainframe Computer in 1993

Memory level Characteristics	Level 0 CPU Registers	Level 1 Cache	Level 2 Main Memory	Level 3 Disk Storage	Level 4 Tape Storage
Device technology	ECL	256K-bit SRAM	4M-bit DRAM	1-Gbyte magnetic disk unit	5-Gbyte magnetic tape unit
Access time, t_i	10 ns	25–40 ns	60–100 ns	12–20 ms	2–20 min (search time)
Capacity, s_i (in bytes)	512 bytes	128 Kbytes	512 Mbytes	60–228 Gbytes	512 Gbytes–2 Tbytes
Cost, c_i (in cents/KB)	18,000	72	5.6	0.23	0.01
Bandwidth, b_i (in MB/s)	400–800	250–400	80–133	3–5	0.18–0.23
Unit of transfer, x_i	4–8 bytes per word	32 bytes per block	0.5–1 Kbytes per page	5–512 Kbytes per file	Backup storage
Allocation management	Compiler assignment	Hardware control	Operating system	Operating system/user	Operating system/user




PROPERTIES OF MEMORY HIERARCHY

INTRODUCTION

- Information stored in memory hierarchy (M1,M2,M3...Mn) satisfies following properties:-
 - ▣ **1. Inclusion property**
 - ▣ **2. Coherence property**
 - ▣ **3. Locality property**
- Cache is considered as innermost level **M1**
- This communicates with CPU registers directly
- Outer most level **Mn** contains all the information words stored



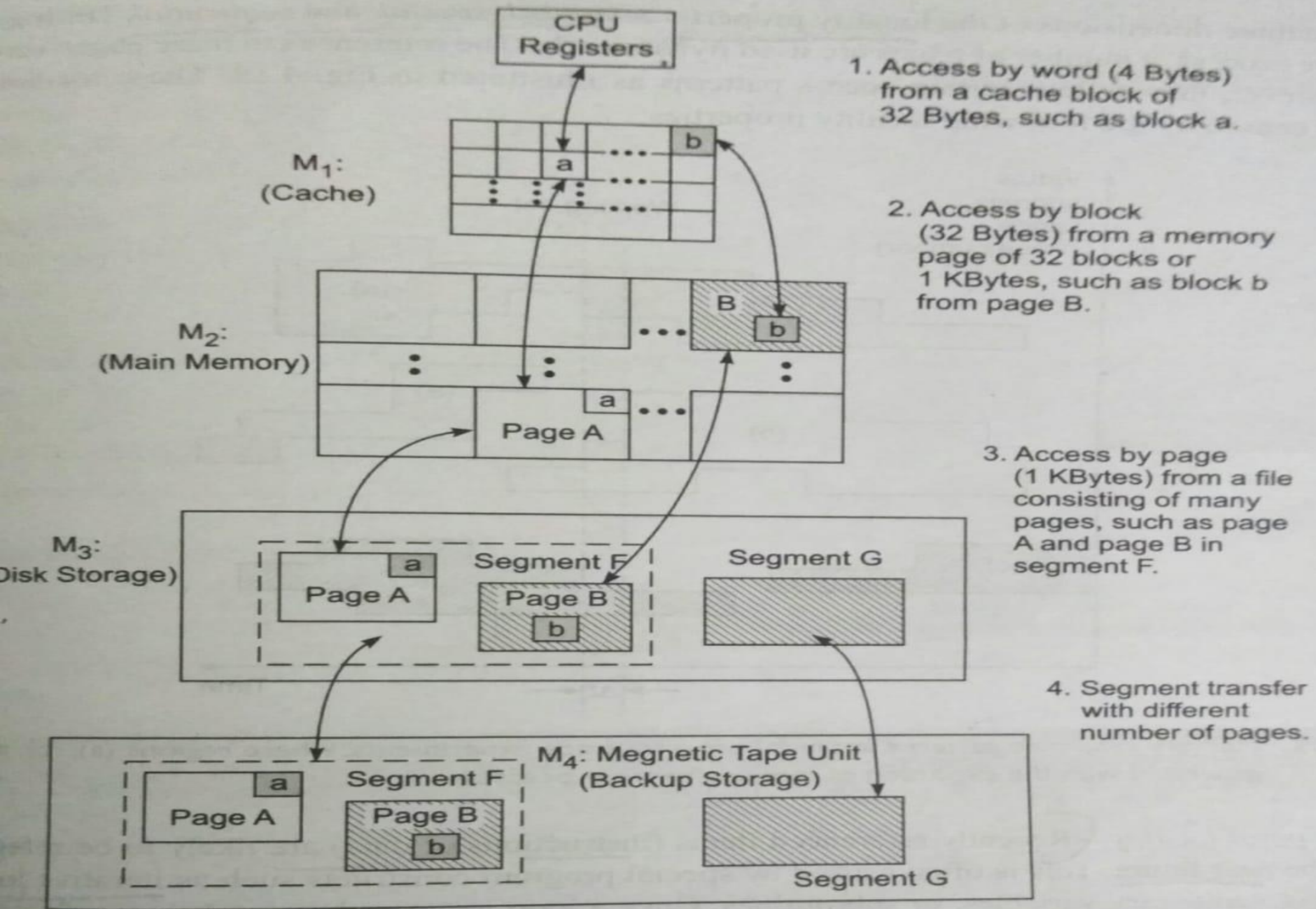
1. INCLUSION PROPERTY

- This property implies that M_1 subset of M_2 subset of M_3 subset of M_n
 - **Inclusion relationship** implies that all information items are originally stored in outermost **level M_n**
 - All the information will be found in highest level which is backup storage
 - During processing, subsets of M_n is copied to M_{n-1}
 - Subset of M_{n-1} are copied to M_{n-2} and so on
 - If an information is found in M_i , then copies of the same word is found in all upper levels, $M_{i+1}, M_{i+2} \dots M_n$
 - However word in M_{i+1} may not be found in M_i
 - **Word miss** in $M_i \rightarrow$ word is missing from all its lower levels
- 

UNITS OF DATA TRANSFER AT EACH LEVEL

- Unit of data transfer between CPU & Cache → words
 - Word → 4 or 8 bytes
- Cache(M1) is divided into cache blocks
 - Each block → 32bytes
 - Unit of transfer between Cache & Main memory → blocks
- Main memory is divided into pages
 - Each page → 4 Kb
 - Unit of data transfer between Main memory & Disks → pages
- Disk memory is organized as segments
 - Size of segment varies with user needs
 - Unit of data transfer between Disk & Backup storage → segments





4.18 The inclusion property and data transfers between adjacent levels of a memory hierarchy

2. COHERENCE PROPERTY

- This property states that, copies of same information item at successive memory levels must be **consistent**
- If a word is modified in cache, the copies of that word must be updated immediately or eventually at all higher levels
- 2 strategies for maintaining coherence
 - Write through (WT)
 - Write back (WB)



○ Write through

- This strategy demands immediate update in M_{i+1} , if a word is modified in M_i for $i=1,2\dots n-1$

○ Write back

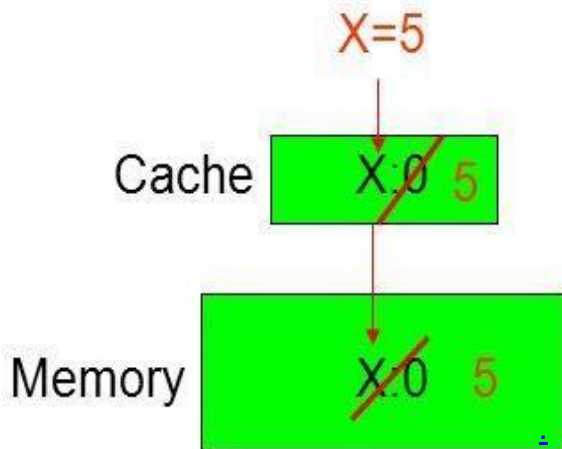
- It delays the update in M_{i+1} until the word being modified in M_i is replaced or removed from M_i



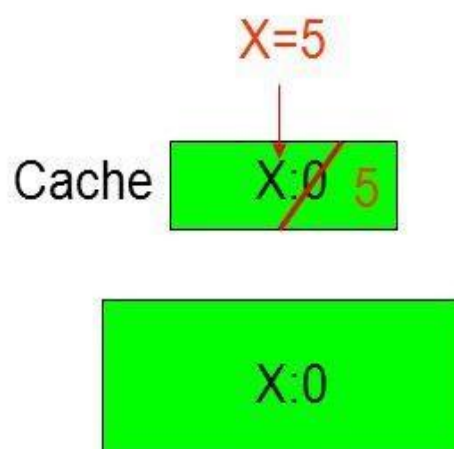
Write-through vs. write-back

- Write-through
 - Every write from every processor goes to shared bus and memory
 - Easy protocol
 - Write-through unpopular for SMPs
- Write-back
 - absorb most writes as cache hits
 - Write hits don't go on bus
 - But now how do we ensure write propagation and serialization?
 - Need more sophisticated protocols

WRITE-THROUGH



WRITE-BACK



3. LOCALITY OF REFERENCES

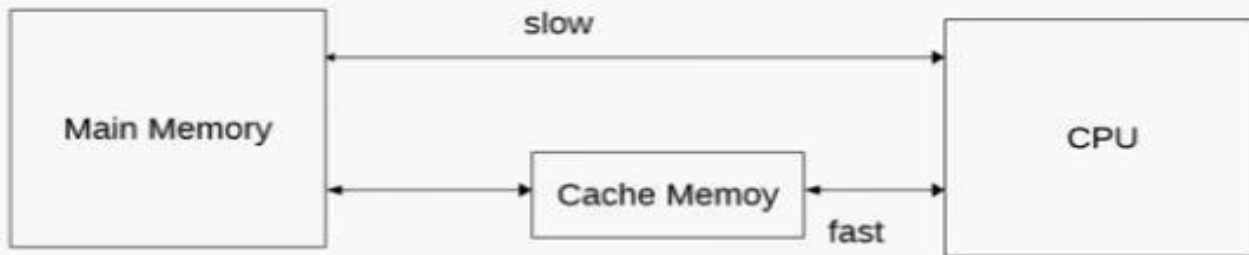
- Locality of reference refers to a phenomenon in which a computer program tends to access same set of memory locations for a particular time period.
- This property of locality of reference is mainly shown by loops and subroutine calls in a program.

LOCALITY OF REFERENCES

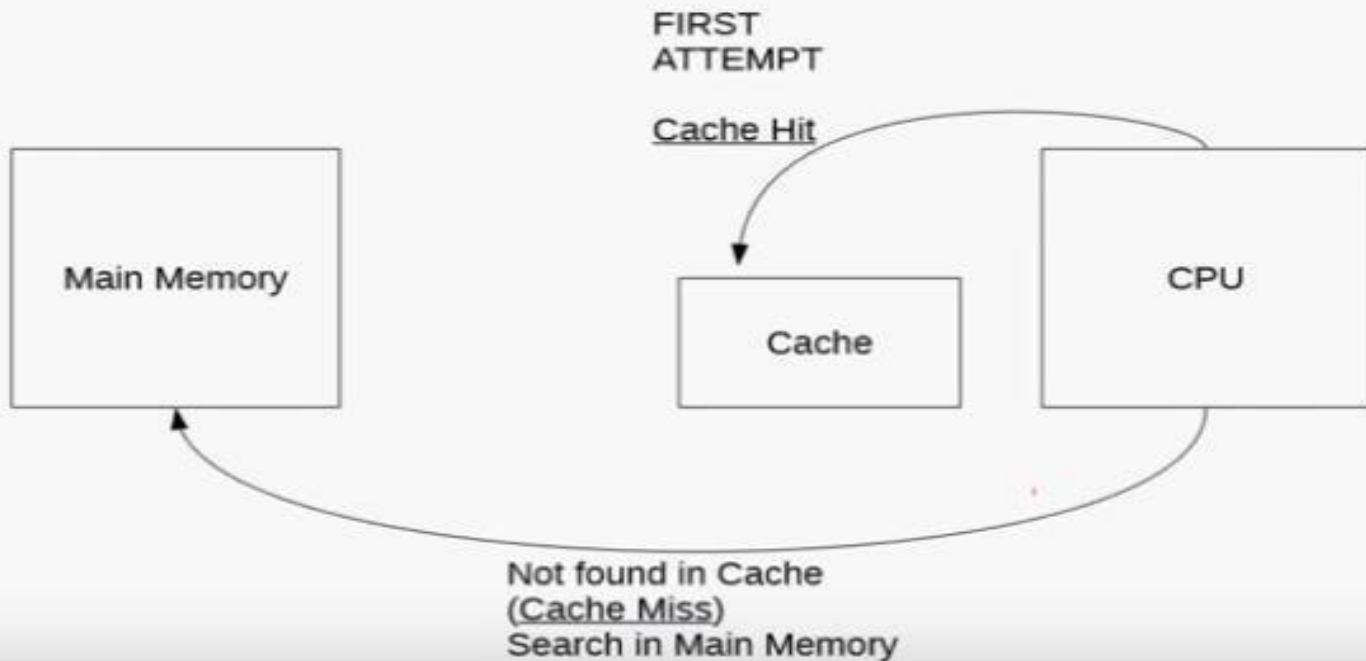
- In case of **Loops** in program CPU repeatedly refers to set of instructions that constitute the loop.
- In case of **subroutine calls**, everytime set of instructions are fetched from memory.
- References to **data items** also get localized that means same data item is referenced again and again.



LOCALITY OF REFERENCES



LOCALITY OF REFERENCES



3. LOCALITY OF REFERENCES

- Memory hierarchy is developed based on locality of reference
- Memory access tends to cluster in certain regions in time, space and ordering
 - Program spend 90% of execution time only in 10% of code
 - Eg: innermost loop of a nested loop operation
- 3 dimensions of locality property
 - [a] Temporal locality
 - [b] Spatial locality

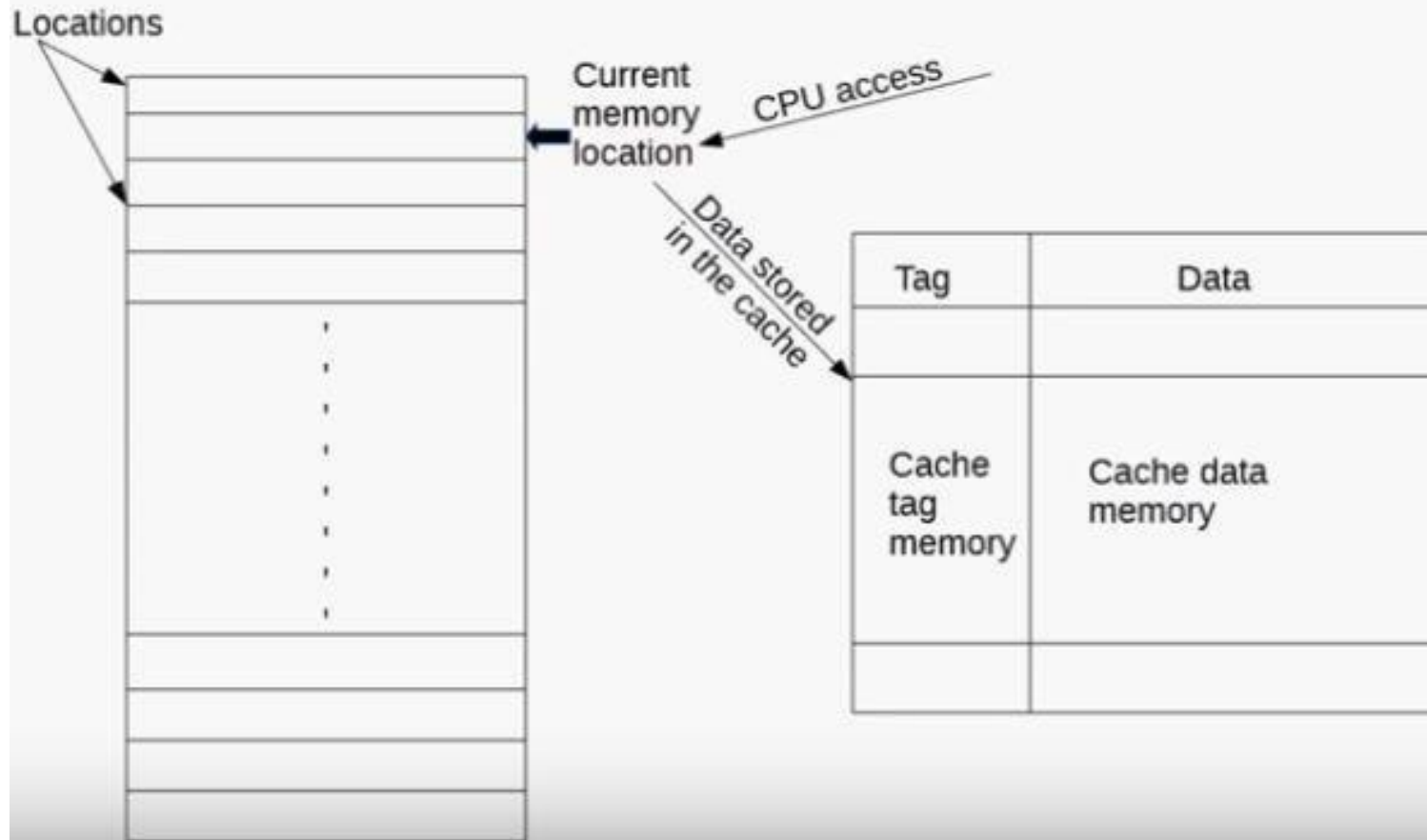


A. TEMPORAL LOCALITY

- Recently referenced items are likely to be referenced again in near future
- This is caused due to program constructs like:-
 - ▣ Iterative loops
 - ▣ Subroutines
- When a loop is encountered or a subroutine is called, a small code segment is referenced repeatedly
- Temporal locality tends to cluster the access in recently used areas



Temporal Locality



Main memory

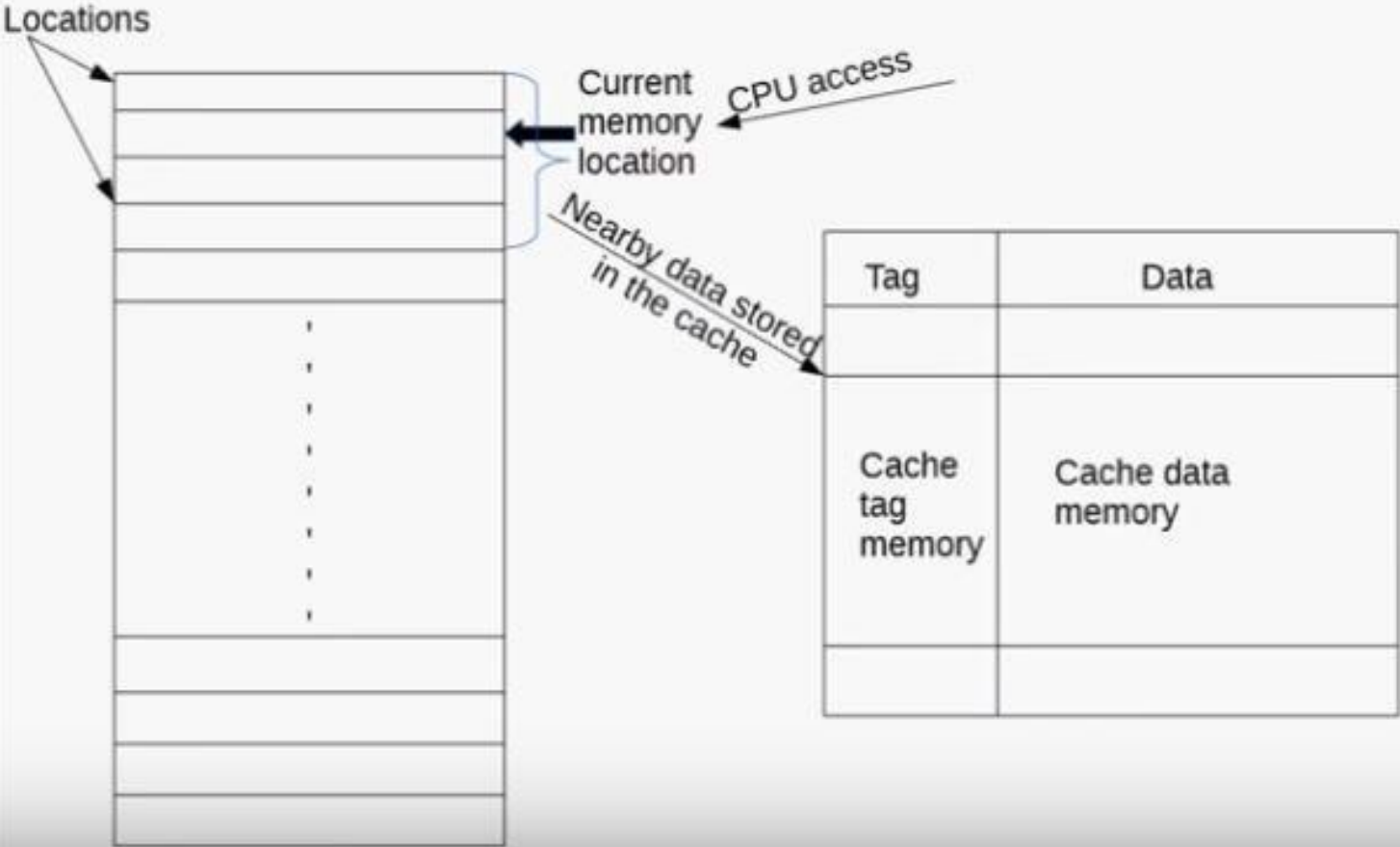
Cache memory

SPATIAL LOCALITY

- Instruction or data near to **main memory** location may be needed soon
 - Eg: operations on arrays
- It involves access of certain clustered area in the address space



Spatial Locality



MEMORY CAPACITY PLANNING

- Performance of a memory hierarchy is determined by **effective access time T_{eff}**
- T_{eff} depends on
 - hit ratio
 - access frequencies of successive levels



HIT RATIO

- It is a concept defined for any 2 adjacent level of memory hierarchy
- When an item is found in $M_i \rightarrow$ **hit**
- Otherwise it is a miss
- Hit ratio H_i at M_i is the probability that an item is found at M_i
- Miss ratio is defined as **$1-H_i$**



ACCESS FREQUENCY

- Access frequency to M_i is defined as
 - the probability of successfully accessing M_i , when there are $i-1$ misses at lower levels and a hit at M_i
- $f_i = (1-h_1)(1-h_2)\dots(1-h_{i-1})h_i$
- Access frequency decreases from lower to high levels
- $f_1 \gg f_2 \gg f_3 \gg \dots \gg f_n$
- This implies that **inner levels** of memory are accessed more often than outer levels



EFFECTIVE ACCESS TIME

- Our aim is to achieve **high hit ratio** at Mi
- Coz when a Miss occur, penalty must be paid to access higher level of memory
 - Misses in cache → block misses
 - Misses in main memory → page fault
- Time penalty for page fault is longer than block miss
 - Because $t_1 < t_2 < t_3$
- **Cache miss** is 2 to 4 times costly than **cache hit**
- **Page fault** is 1000 to 10000 times costly than **page hit**



○ $T_{eff} = \sum_{i=1}^n f_i \cdot t_i$

$$= h_1 t_1 + (1-h_1)h_2 t_2 + (1-h_1)(1-h_2)h_3 t_3 + \dots + (1-h_1)(1-h_2)\dots(1-h_{n-1})t_n$$

T_{eff} depends on **program behavior** and **memory choices**



HIERARCHY OPTIMIZATION

- Total cost of memory hierarchy is C_{total}
- $C_{\text{total}} = \sum_{i=1}^n c_i \cdot s_i$
- This implies that cost is estimated over n levels
- c_i and s_i depends on t_i at each levels



PROBLEM

- Consider the design of a 3 level memory hierarchy with following features

Memory level	Access time	Capacity	Cost/kbyte
Cache	$t_1 = 25 \text{ ns}$	$s_1 = 512 \text{ Kbytes}$	$c_1 = \$1.25$
Main memory	$t_2 = \text{unknown}$	$s_2 = 32 \text{ Mbytes}$	$c_2 = \$.2$
Disk <u>array</u>	<u>$t_3 = 4 \text{ ms}$</u>	<u>$s_3 = \text{unknown}$</u>	<u>$c_3 = \\$0.0002$</u>

- Aim is to achieve effective memory accesstime $T_{\text{eff}} = 850 \text{ ns}$
- Cache hit ratio $h_1 = 0.98$
- Main memory Hit ratio $h_2 = 0.99$
- Total cost is upper bounded by \$15000
- Calculate the unknown specifications based on the given conditions



○ $C_{\text{total}} = c_1 \cdot s_1 + c_2 \cdot s_2 + c_3 \cdot s_3 \leq 1500$

$$15000 = 1.25 \cdot 512 + .2 \cdot 32000 + .0002 s_3$$

$$15000 = 640 + 6400 + .0002 s_3$$

$$\underline{.0002 s_3 = 15000 - 640 - 6400}$$

$$\underline{s_3 = 7960 / .0002}$$

$$= 39800000 \times 10^{-6}$$

$$= 39.8$$



- $T_{eff} = h_1 t_1 + (1-h_1)h_2 t_2 + (1-h_1)(1-h_2)h_3 t_3$
- $T_{eff} = h_1 t_1 + (1-h_1)h_2 t_2 + (1-h_1)(1-h_2)h_3 t_3 \leq 850 \text{ ns}$
- $850 \times 10^{-9} = .98 * 25 \times 10^{-9} + .02 * .99 * t_2 + .02 * .01 * 1 * 4 * 10^{-3}$
- $850 \times 10^{-9} = \underline{24.5 \times 10^{-9} + .0198 t_2 + .0008 \times 10^{-3}}$
- $\underline{.0198 t_2 = 850 \times 10^{-9} - 24.5 \times 10^{-9} - .0008 \times 10^{-3}}$

$$= 825.5 \times 10^{-9} - .0008 \times 10^{-3}$$

$$= 825.5 \times 10^{-9} - 800 \times 10^{-9}$$

$$= 25.5 \times 10^{-9}$$

$$= 25.5 \times 10^{-9} / .0198$$

$$t_2 = 1287 \times 10^{-9}$$

