Advanced Computer Architecture

17CS72

MODULE-2

Hardware Technologies

Book: "Advanced Computer Architecture – Parallelism, Scalability, Programmability", Hwang & Jotwani

https://hemanthrajhemu.github.io

MODULE-2

Syllabus:

MODULE	Hardware Technologies:	
2	**Hardware Technologies:** Processors	2
	Hardware Technologies: Memory Hierarchy	2
	Advanced Processor Technology	2
	Superscalar and Vector Processors	2
	Memory Hierarchy Technology, Virtual Memory Technology.	2

https://hemanthrajhemu.github.io

Hardware Technologies

https://hemanthrajhemu.github.io

Processors and Memory Hierarchy

- Advanced Processor Technology
  - Design Space of Processors
  - Instruction-Set Architectures
  - CISC Scalar Processors
  - RISC Scalar Processors
- Superscalar and Vector Processors
  - Superscalar Processors
  - VLIW Architecture
  - Vector and Symbolic Processors

https://hemanthrajhemu.github.io

Advanced Processor Technology

Design Space of Processors

- Various processor families can be mapped onto space of clock rate versus cycles per instruction
- Mapping processor families onto a coordinated space of

  clock rate vs cycles per instruction (CPI)
- Trends:-
  - Clock rates are moving from low to high
    - Implementation technology
  - Lowering CPI rate
  - Hardware and software approaches

Advanced Processor Technology

Design Space of Processors

- Various processor families can be mapped onto
  - Space of clock rate versus cycles per instruction

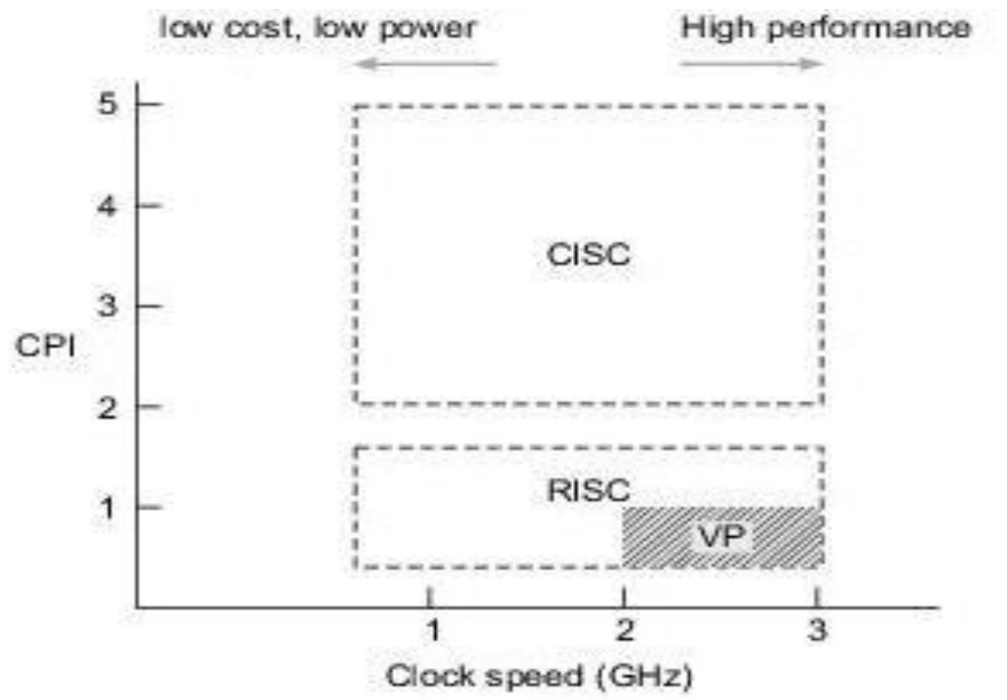


Fig. 4.1 CPI versus processor clock speed of major categories of processors

Advanced Processor Technology

Design Space of Processors

- CISC (Complex Instruction Set Computation) ➔
  - The CPI of different CISC instructions varies from 1 to 20.
  - Conventional processors like the Intel Pentium, IBM 390,etc

Advanced Processor Technology

Design Space of Processors

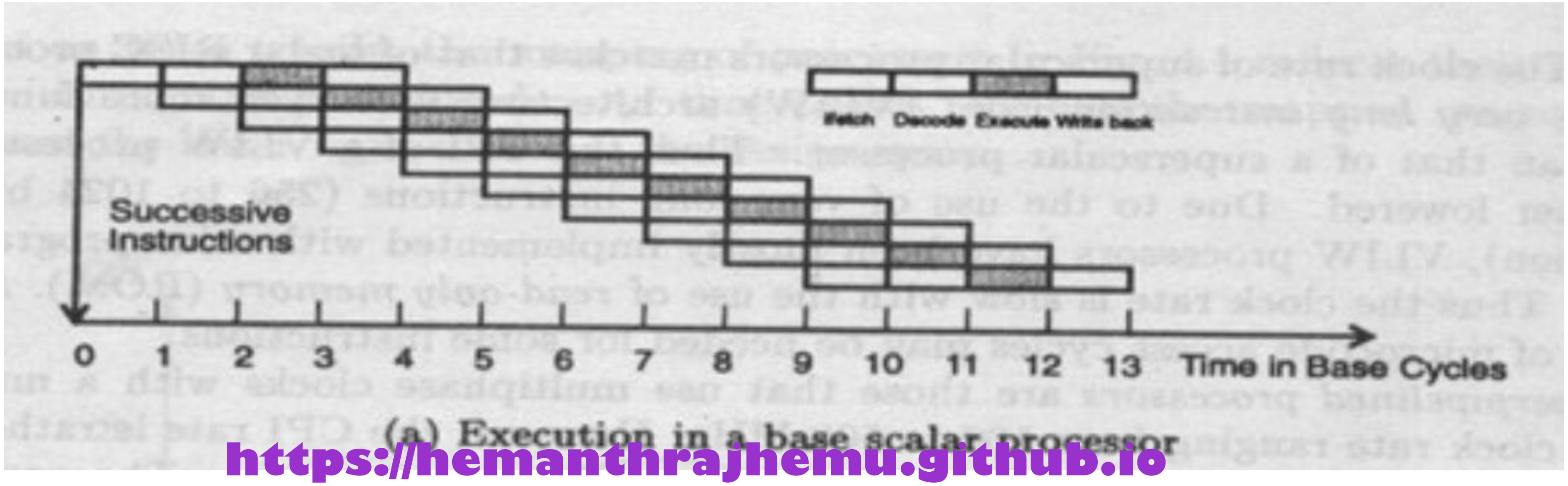
- **RISC (Reduced Instruction Set Computation):➔**
  - Processors include SPARC, MIPS, ARM. etc.
  - The average CPI between one and two cycles (With the use of pipelines)
  - Few Subclasses of RICS:➔
    - **Superscalar** processors allow multiple instructions to be issued simultaneously during each cycle.
      - Thus effective CPI less than scalar RISC
    - **Vector supercomputers**
      - Use multiple function units for execution.
    - **VLIW architecture** (Very Long Instruction Word) can in theory use even more functional units
      - Thus effective CPI less than superscalar architecture.
      - Intel's i860 RISC processor had VLIW architecture.

Advanced Processor Technology

Instruction Pipeline

Pipeline cycle - four phases: **fetch, decode, execute, and write-back**.

- **Base scalar processor:**➔ A machine with
  - One instruction issued per cycle,
  - One-cycle latency for a simple operation, and
  - One-cycle latency between instruction issues.



(a) Execution in a base scalar processor

Advanced Processor Technology

Instruction Pipeline

Basic Definitions:

– **Instruction pipeline cycle:** clock period of instruction pipeline

– **Instruction issue latency:** cycles between two adjacent issue

– **Instruction issue rate:** number of instruction issue per cycle

– **Simple operation latency**: integer adds, load, store, branch, move

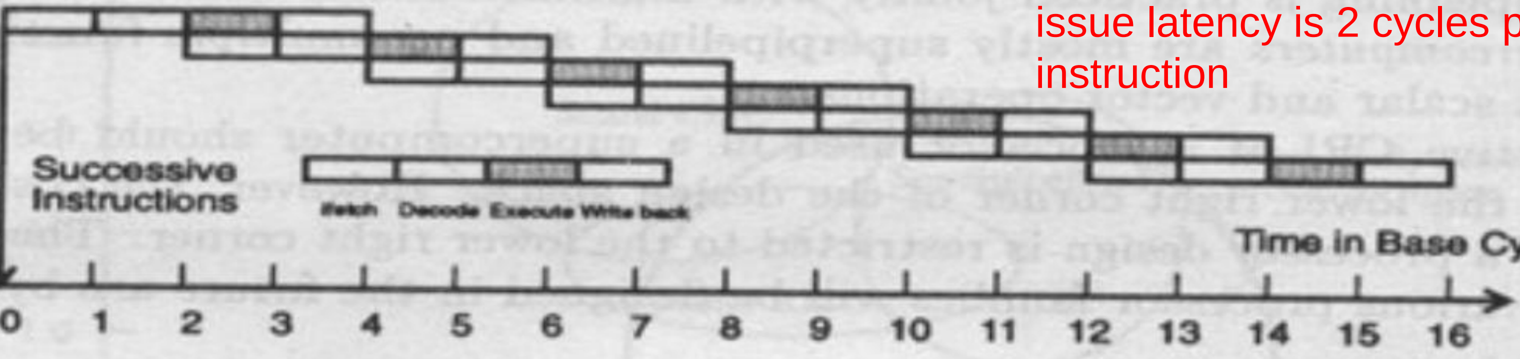
– **Resource conflict:** same functional unit at same time

https://hemanthrajhemu.github.io

Advanced Processor Technology

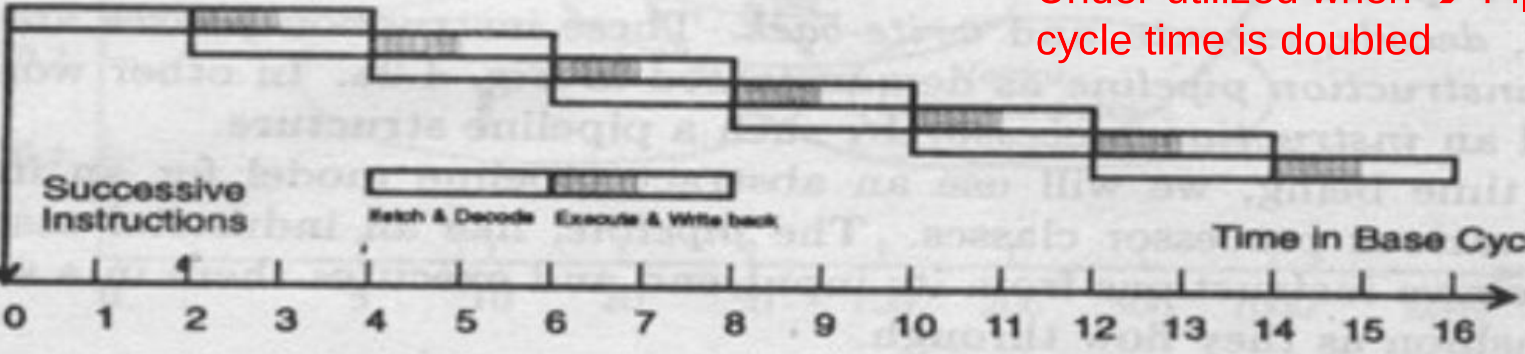
Instruction Pipeline

Under-utilized when ➔ Instruction issue latency is 2 cycles per instruction



(b) Underpipelined with two cycles per instruction issue

Under-utilized when ➔ Pipeline cycle time is doubled



(c) Underpipelined with twice the base cycle

https://hemanthrajhemu.github.io

Advanced Processor Technology

Instruction Pipeline

Control unit

- hardwired
- micro-coded

- Program status word (**PSW**)

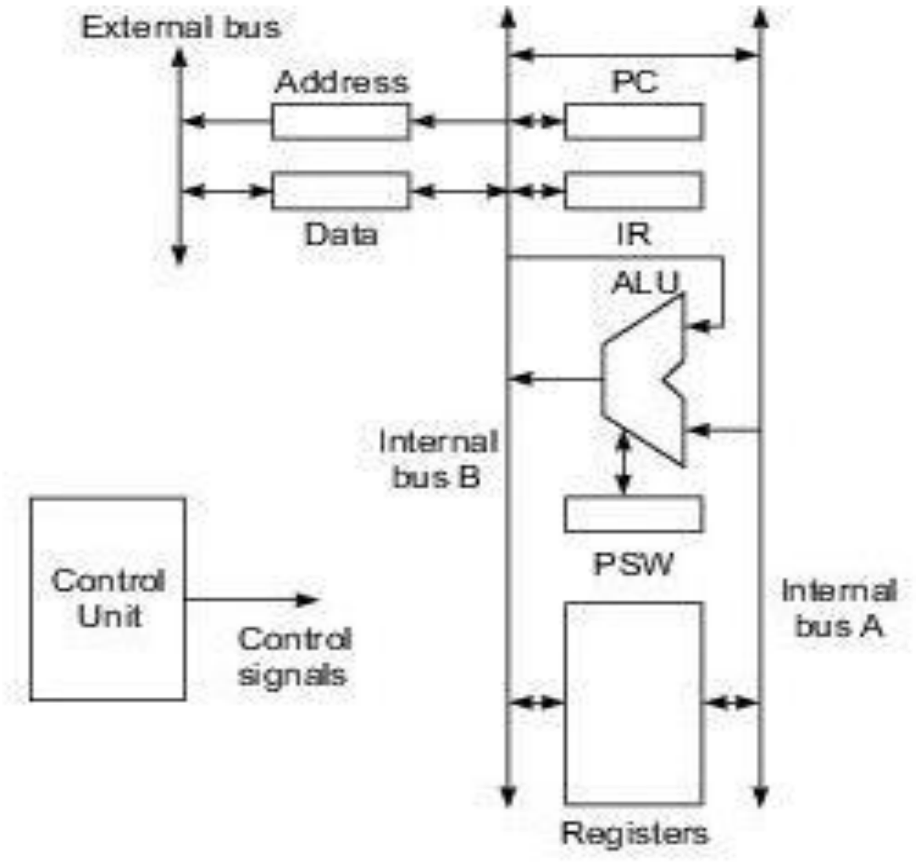


Fig. 4.3 Data path architecture and control unit of a scalar processor

Instruction Set Architecture

- The instruction set of a computer specifics➔
  - 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 specifications
  - Flow control mechanisms

Instruction Set Architecture

- ***Complex Instruction Set:***
  - 120 to 350 instructions using variable instruction/data formats
  - a large number of memory reference operations based on more than a dozen addressing modes.
  - 8 to 24 general-purpose registers and many memory based operations
  - Many HLL statements are directly implemented in Hardware

https://hemanthrajhemu.github.io

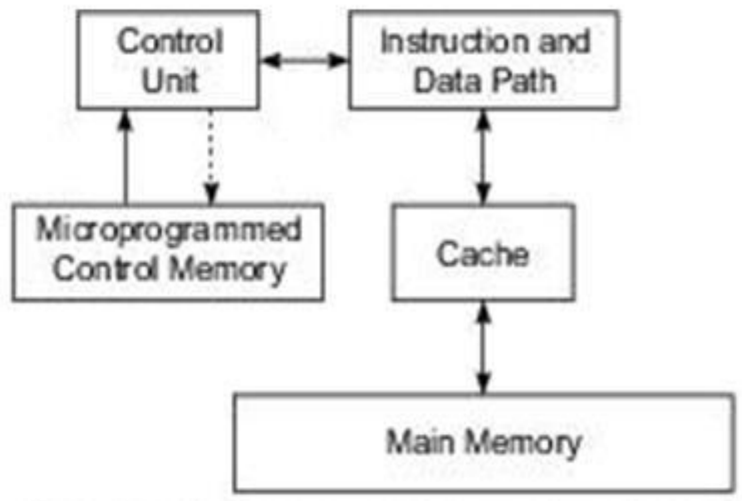
Instruction Set Architecture

- ***Reduced Instruction Set:***
  - less than 100 instructions with a fixed instruction format [32 bits].
  - 3 to 5 simple addressing modes are used.
  - Most instructions are register-based
  - Memory access is done by load/store instructions
  - A large register file [at least 32] is used to improve fast context switching.
  - most instructions execute in one cycle.

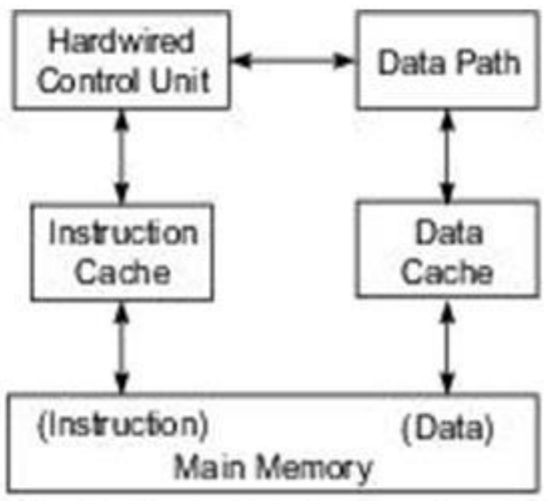
https://hemanthrajhemu.github.io

Instruction Set Architecture

Architectural Distinction:



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

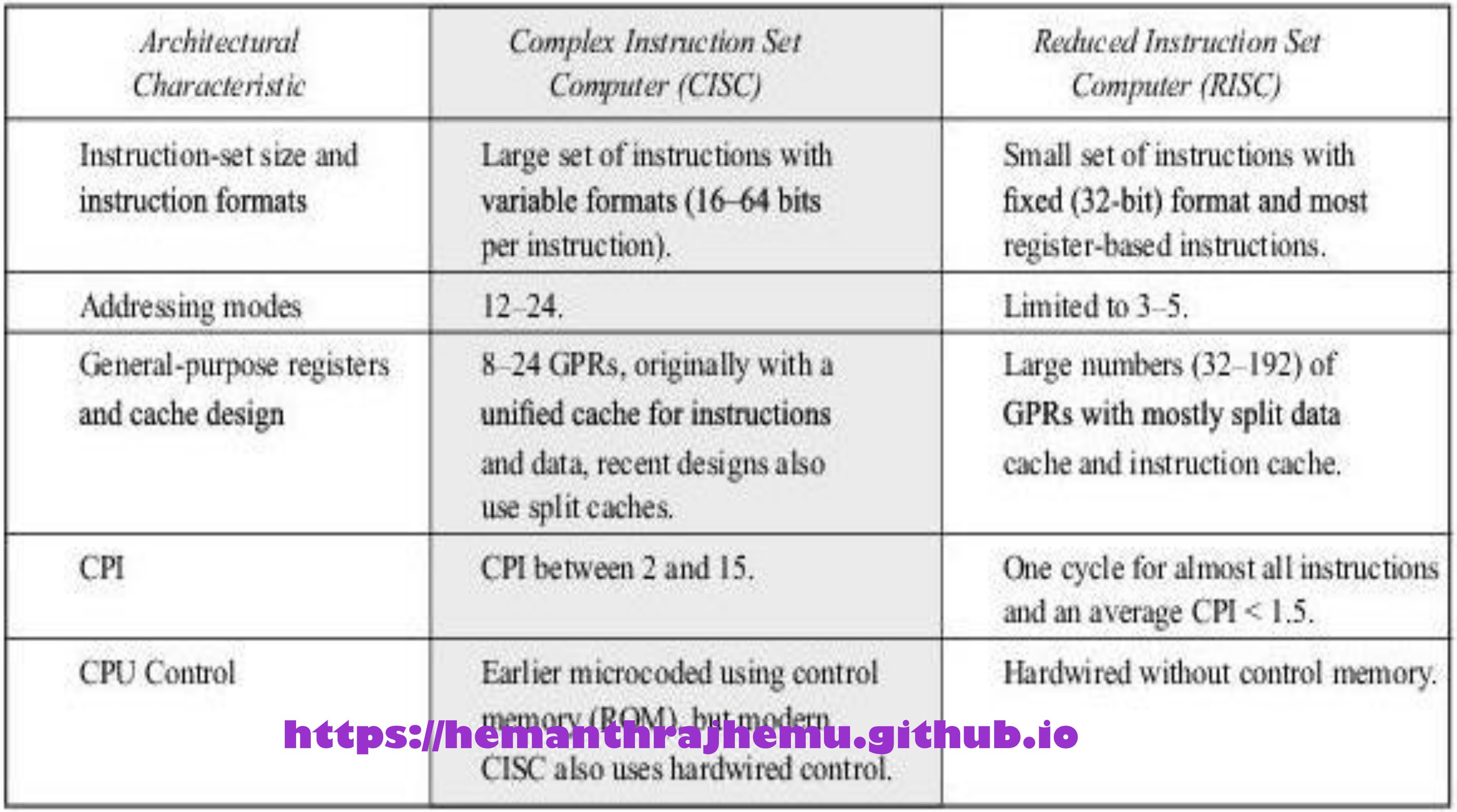


(b) The RISC architecture with hardwired control and split instruction cache and data cache

- The **hardwired control unit** does not require a **control** memory as here; the **control** signal is generated by hardware.
- The **microprogrammed control unit** requires the **control** memory as the microprograms which are responsible for generating the **control** signals are stored in **control** memory

Instruction Set Architecture

Table 4.1 *Characteristics of Typical CISC and RISC Architectures*

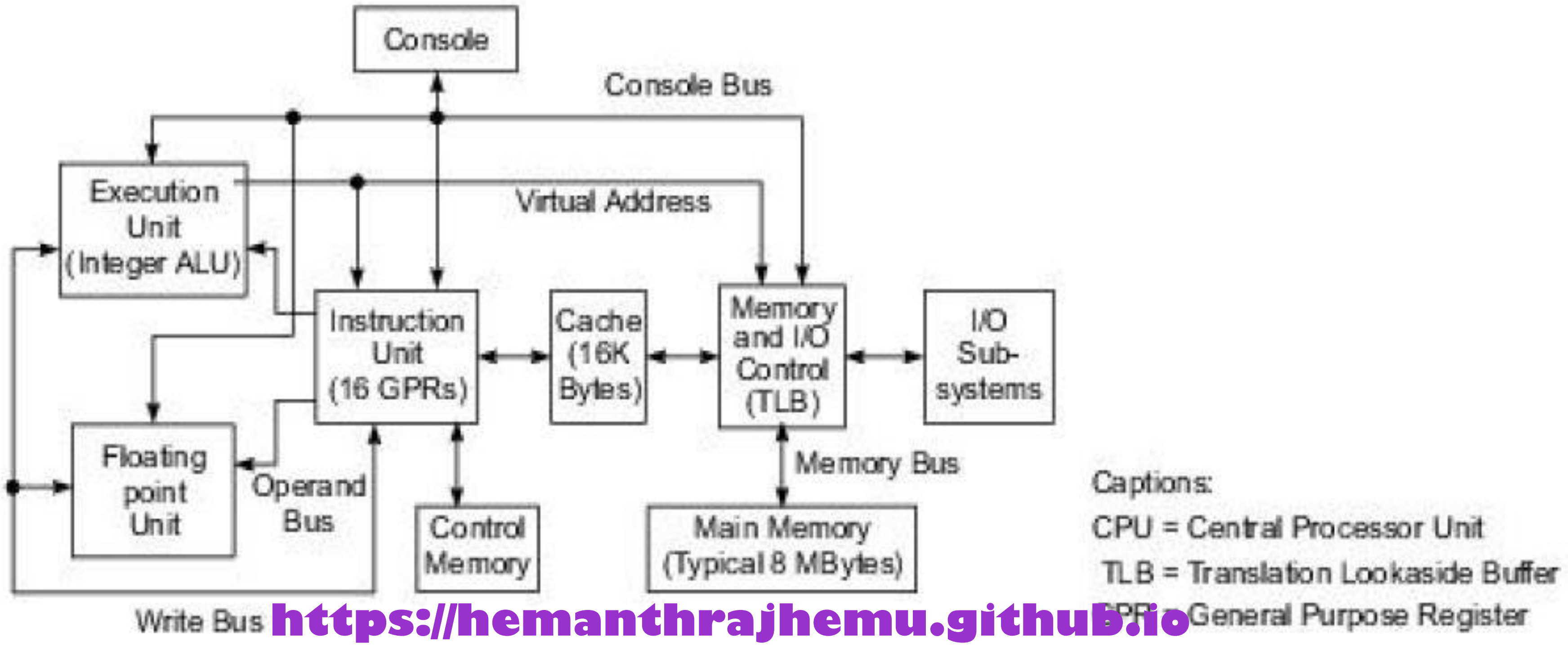


Architectural Characteristic	*Complex Instruction Set Computer (CISC)*	*Reduced Instruction Set Computer (RISC)*
Instruction-set size and instruction formats	Large set of instructions with variable formats (16–64 bits per instruction).	Small set of instructions with fixed (32-bit) format and most register-based 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.

Instruction Set Architecture

CISC Scalar Processors:

- Executes with scalar data
- The simplest scalar processor executes integer instructions using fixed-point operands.
- Modern systems also used pipeline (but under utilized)

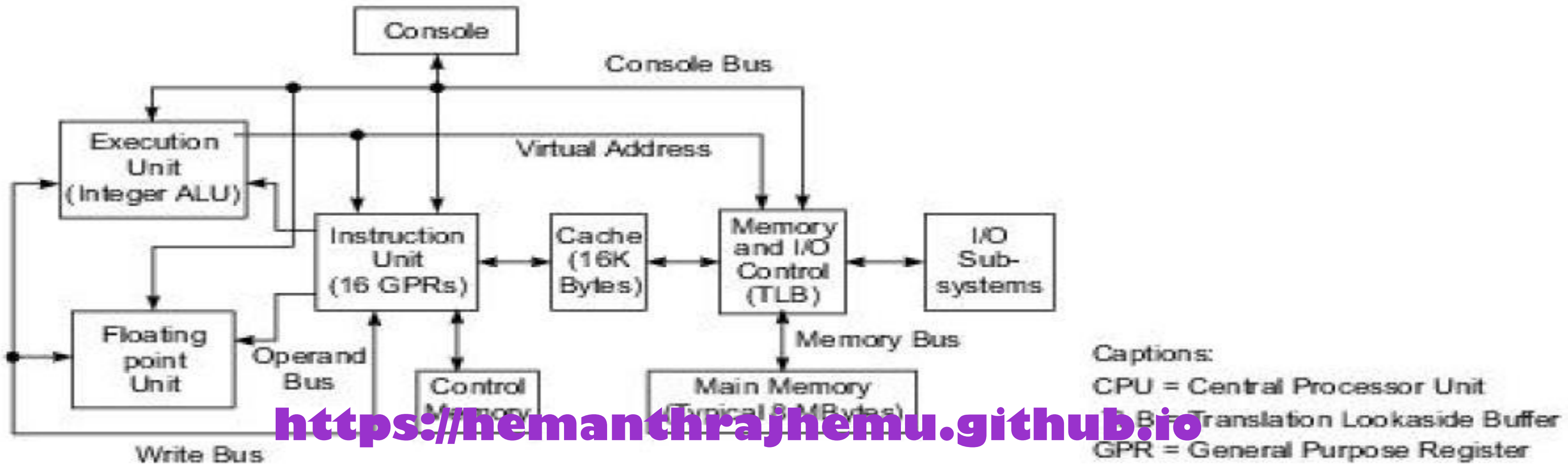

Console
Console Bus
Execution Unit (Integer ALU)
Virtual Address
Instruction Unit (16 GPRs)
Cache (16K Bytes)
Memory and I/O Control (TLB)
I/O Sub-systems
Floating point Unit
Operand Bus
Control Memory
Memory Bus
Main Memory (Typical 8 MBytes)
Write Bus
Captions:
CPU = Central Processor Unit
TLB = Translation Lookaside Buffer
GPR = General Purpose Register

ig. 4.5 The VAX 8600 CPU, a typical CISC processor architecture (Courtesy of Digital Equipment Corporation, 1985)

Instruction Set Architecture

CISC Scalar Processors:

- The instruction set contained about 300 instructions with 20 different addressing modes.
- Two functional units for concurrent execution of integer and floating-point instructions unit.
- The unified cache was used for holding both instructions and data.
- There were 16 GPRs in the instruction unit
- Instruction pipelining was done with 6 stages
- TLB-For fast generation of physical address from virtual address
- Performance dependence on catch hit ratio.

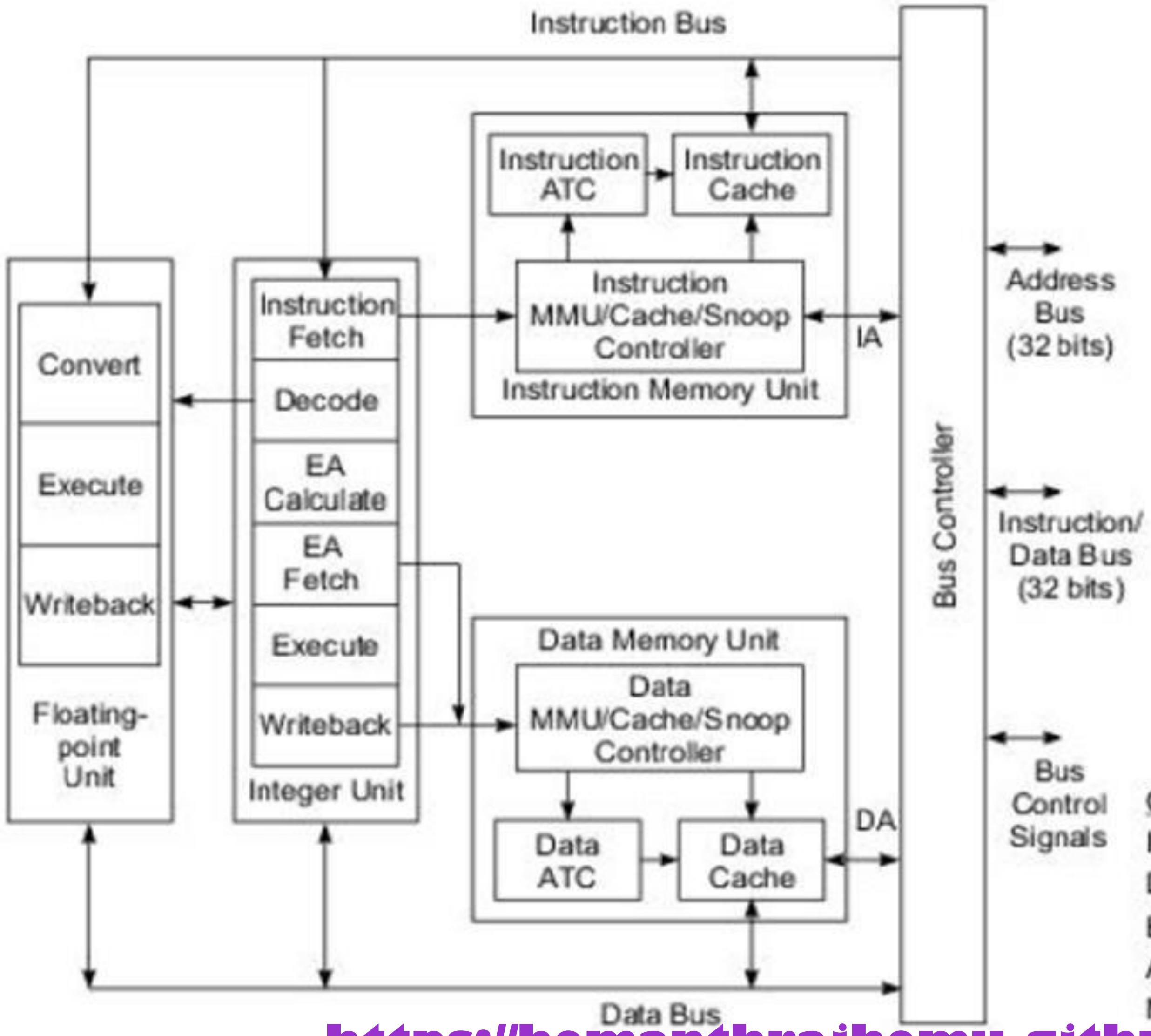

Console
Console Bus
Execution Unit (Integer ALU)
Virtual Address
Instruction Unit (16 GPRs)
Cache (16K Bytes)
Memory and I/O Control (TLB)
I/O Sub-systems
Floating point Unit
Operand Bus
Control Memory
Main Memory (Typical 8 MBytes)
Memory Bus
Write Bus
Captions:
CPU = Central Processor Unit
TLB = Translation Lookaside Buffer
GPR = General Purpose Register

ig. 4.5 The VAX 8600 CPU, a typical CISC processor architecture (Courtesy of Digital Equipment Corporation, 1985)

Table 4.2 *Representative CISC Scalar Processors of year 1990*

Feature	*Intel i486*	*Motorola MC68040*	*NS 32532*
Instruction-set size and word length	157 instructions, 32 bits.	113 instructions, 32 bits.	63 instructions, 32 bits.
Addressing modes	12	18	9
Integer unit and GPRs	32-bit ALU with 8 registers.	32-bit ALU with 16 registers.	32-bit ALU with 8 registers.
On-chip cache(s) and MMUs	8-KB unified cache for both code and data. with separate MMUs.	4-KB code cache 4-KB data cache	512-B code cache 1-KB data cache.
Floating-point unit, registers, and function units	On-chip with 8 FP registers adder, multiplier, shifter.	On-chip with 3 pipeline stages, 8 80-bit FP registers.	Off-chip FPU NS 32381, or WTL 3164.
Pipeline stages	5	6	4
Protection levels	4	2	2
Memory organization and TLB/ATC entries	Segmented paging with 4 KB/page and 32 entries in TLB.	Paging with 4 or 8 KB/page, 64 entries in each ATC.	Paging with 4 KB/page, 64 entries.
Technology, clock rate, packaging, and year introduced	CHMOS IV, 25 MHz, 33 MHz, 1.2M transistors, 168 pins, 1989.	0.8–μm HCMOS, 1.2 M transistors, 20 MHz, 40 MHz, 179 pins, 1990.	1.25–μm CMOS 370K transistors, 30 MHz, 175 pins, 1987.
Claimed performance	[illegible]	[illegible] 30 MIPS at 60 MHz.	15 MIPS at 30 MHz.

The Motorola MC68040 microprocessor architecture



- 32-bit microprocessor
- >1.2 million transistors
- 100 instructions ranging 8 to 80 bits
- 18 addressing modes
- 16GPR
- 4KB data cache
- 4KB instruction cache
- Separate MMU with ATC
- Integer unit- 6 stage pipeline
- Floating unit- 3 stage pipeline
- Snooping logic

Captions:
IA = Instruction Address
DA = Data Address
EA = Effective Address
ATC = Address Translation Cache
MMU = Memory Management Unit

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

Instruction Set Architecture

RISC Scalar Processors:

- RISC is usually called scalar RISC as they issue one instruction per cycle.
- RISC pushes some of the less frequently used instructions to software, making the software complex.
- RISC processors depend heavily on a good compiler because complex HLL instructions are to be converted into primitive low level instructions, which are few in number.
- All these processors use 32-bit instructions.
- The instruction sets consist of 51 to 124 basic instructions.

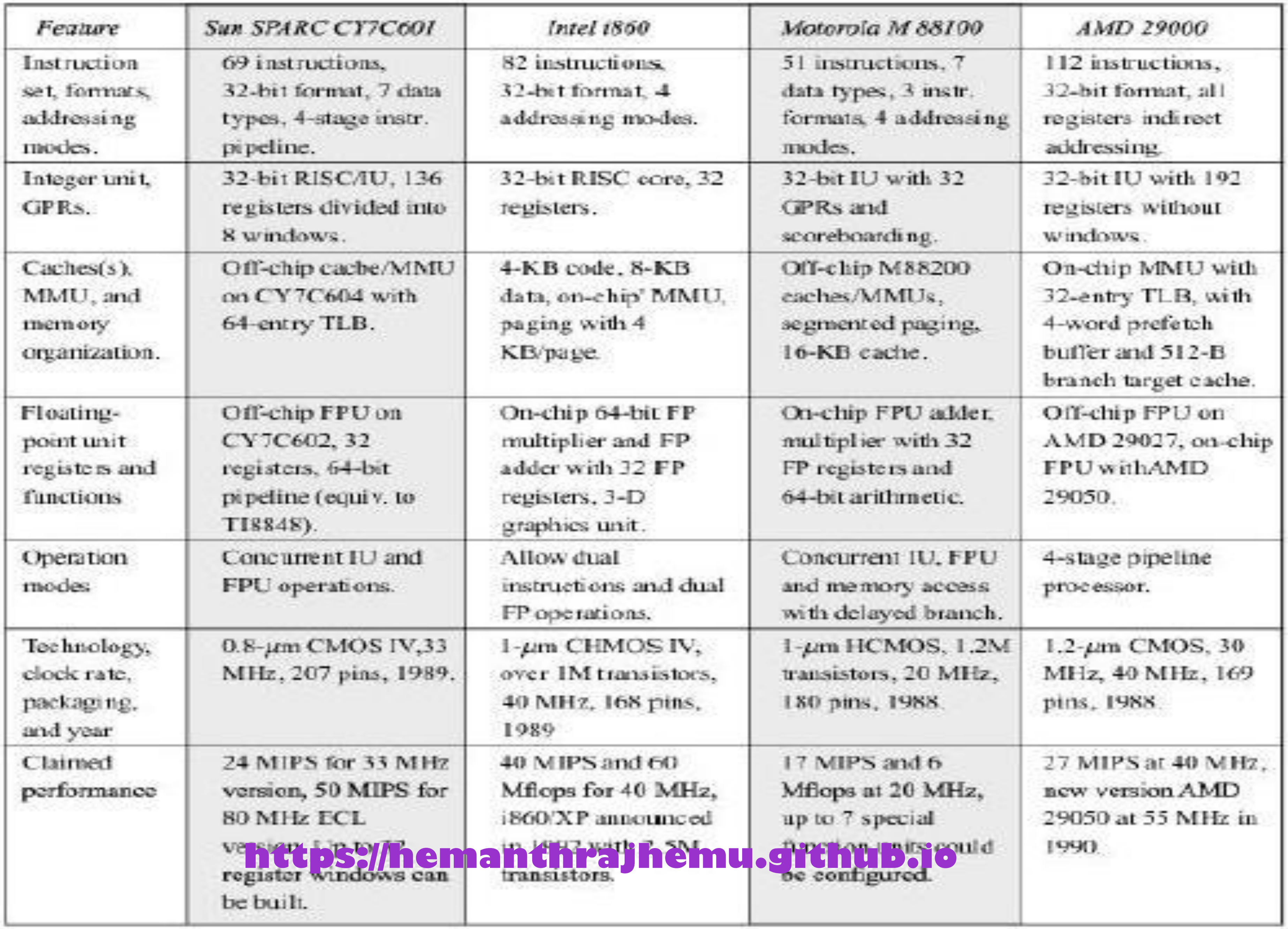
Instruction Set Architecture

RISC Scalar Processors:

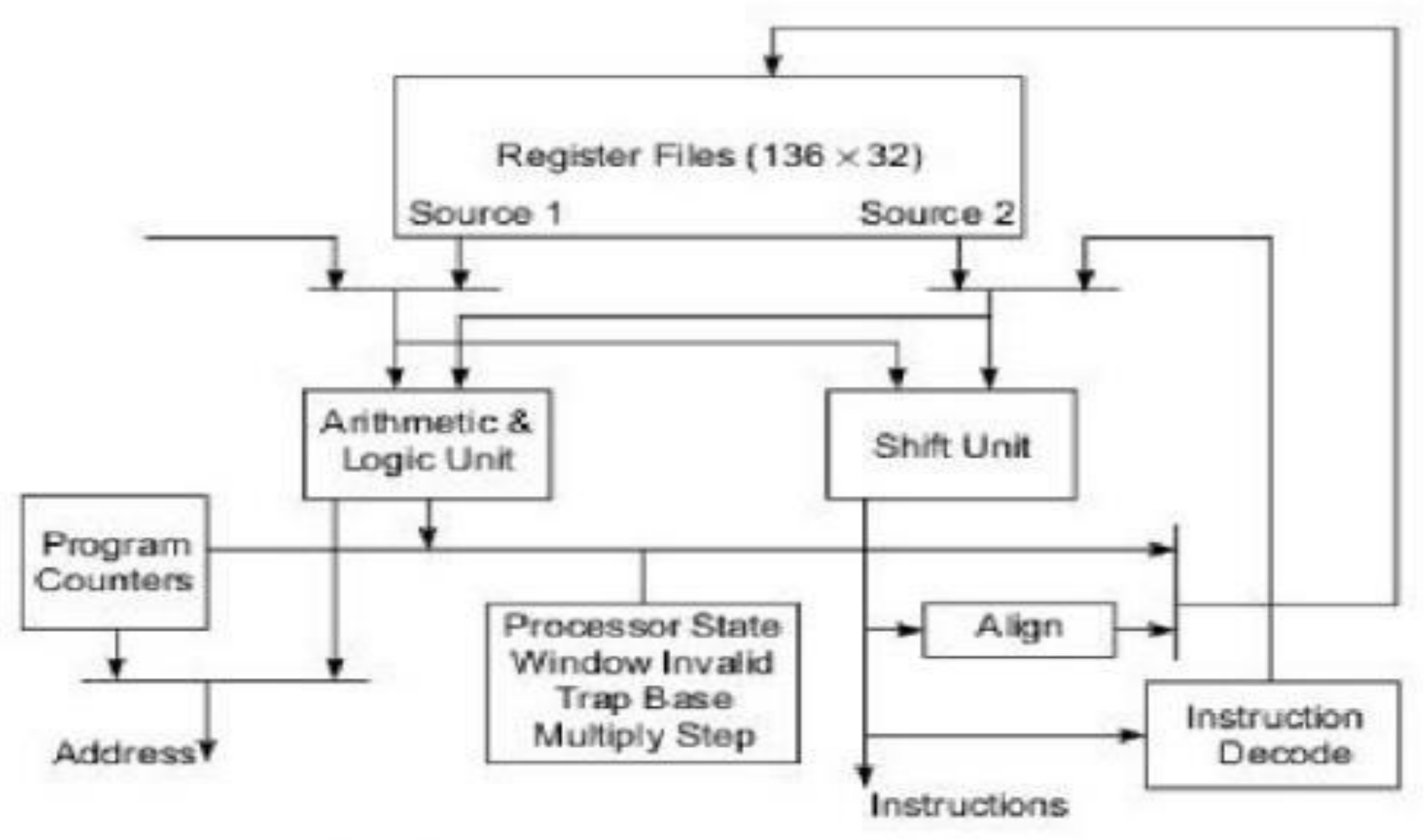
- Four representative RISC-based processors from the year 1990
  - Sun SPARC (Scalable Processor ARChitecture)
  - Intel i860
  - Motorola M88100
  - AMD 29000
- **On-chip floating-point units** are built into the i860 and M88100, while the SPARC and AMD use off-chip floating point units.

https://hemanthrajhemu.github.io

Table 4.3 *Representative RISC Scalar Processors of year 1990*

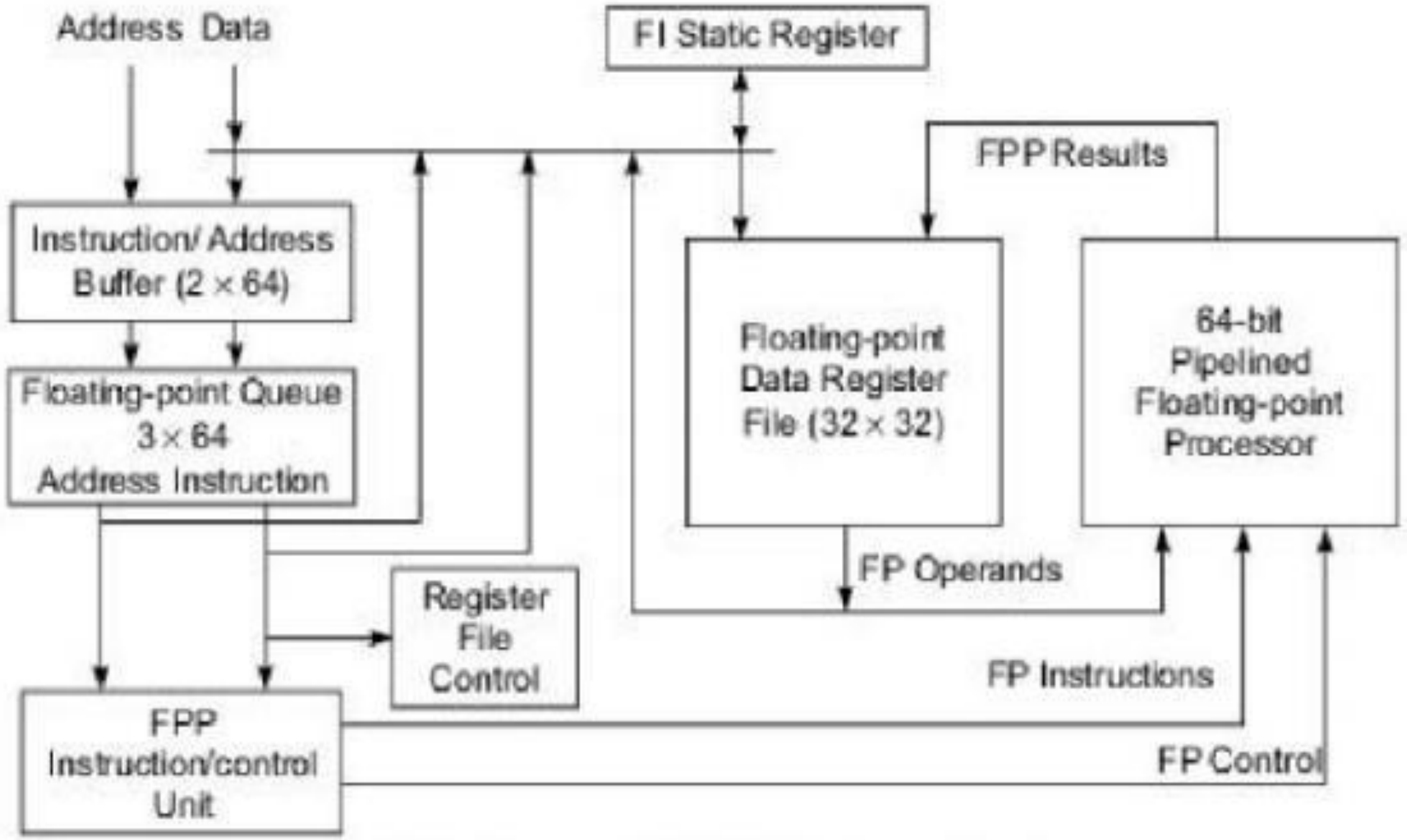


Feature	*Sun SPARC CY7C601*	*Intel i860*	*Motorola M 88100*	*AMD 29000*
Instruction set, formats, addressing modes.	69 instructions, 32-bit format, 7 data types, 4-stage instr. pipeline.	82 instructions, 32-bit format, 4 addressing modes.	51 instructions, 7 data types, 3 instr. formats, 4 addressing modes.	112 instructions, 32-bit format, all registers indirect addressing.
Integer unit, GPRs.	32-bit RISC/IU, 136 registers divided into 8 windows.	32-bit RISC core, 32 registers.	32-bit IU with 32 GPRs and scoreboarding.	32-bit IU with 192 registers without windows.
Caches(s), MMU, and memory organization.	Off-chip cache/MMU on CY7C604 with 64-entry TLB.	4-KB code, 8-KB data, on-chip MMU, paging with 4 KB/page.	Off-chip M88200 caches/MMUs, segmented paging, 16-KB cache.	On-chip MMU with 32-entry TLB, with 4-word prefetch buffer and 512-B branch target cache.
Floating-point unit registers and functions	Off-chip FPU on CY7C602, 32 registers, 64-bit pipeline (equiv. to TI8848).	On-chip 64-bit FP multiplier and FP adder with 32 FP registers, 3-D graphics unit.	On-chip FPU adder, multiplier with 32 FP registers and 64-bit arithmetic.	Off-chip FPU on AMD 29027, on-chip FPU with AMD 29050.
Operation modes	Concurrent IU and FPU operations.	Allow dual instructions and dual FP operations.	Concurrent IU, FPU and memory access with delayed branch.	4-stage pipeline processor.
Technology, clock rate, packaging, and year	0.8-μm CMOS IV, 33 MHz, 207 pins, 1989.	1-μm CHMOS IV, over 1M transistors, 40 MHz, 168 pins, 1989	1-μm HCMOS, 1.2M transistors, 20 MHz, 180 pins, 1988.	1.2-μm CMOS, 30 MHz, 40 MHz, 169 pins, 1988.
Claimed performance	24 MIPS for 33 MHz version, 50 MIPS for 80 MHz ECL version. Up to 32 register windows can be built.	40 MIPS and 60 Mflops for 40 MHz, i860/XP announced in 1992 with 2.5M transistors.	17 MIPS and 6 Mflops at 20 MHz, up to 7 special function units could be configured.	27 MIPS at 40 MHz, new version AMD 29050 at 55 MHz in 1990.



(a) The Cypress CY7C601 SPARC processor

https://hemanthrajhemu.github.io


Address Data
FI Static Register
FPP Results
Instruction/ Address Buffer (2 × 64)
Floating-point Queue 3 × 64 Address Instruction
Floating-point Data Register File (32 × 32)
64-bit Pipelined Floating-point Processor
FP Operands
Register File Control
FP Instructions
FPP Instruction/control Unit
FP Control

(b) The Cypress CY7C602 floating-point unit

Example 1: Sun microsystems SPARC

SPARC stands for Scalable Processor ARChitecture

Floating point unit (FPU) is implemented on a separate coprocessor chip

SPARC processor consists of integer unit (IU) implemented with 2-32 register windows.

SPARC runs each procedure with a set of thirty two 32-bit registers

- Eight of these registers are *global registers* shared by all procedures
- Remaining 24 are window registers which are not shared.

Concept of using overlapped registers is the most important feature introduced

Each register window is divided into 3 eight register sections

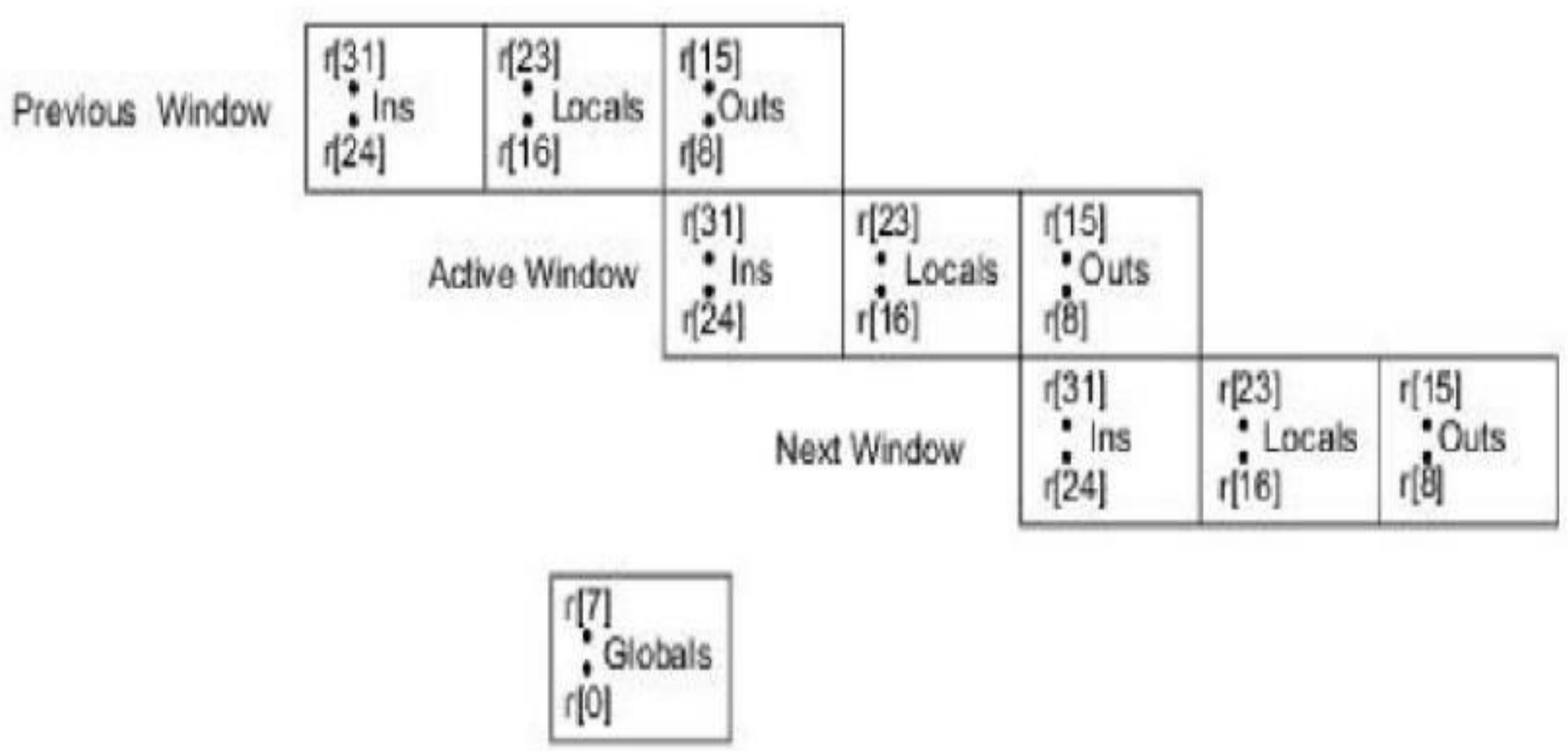
- *Ins (shared)*
  - Values is called procedure
- *Locals*
  - Locally addressable by each procedure
- *Outs (shared)*
  - Values in calling procedure

Current window pointer- points to currently running procedure

Window invalid mask- to mark invalid

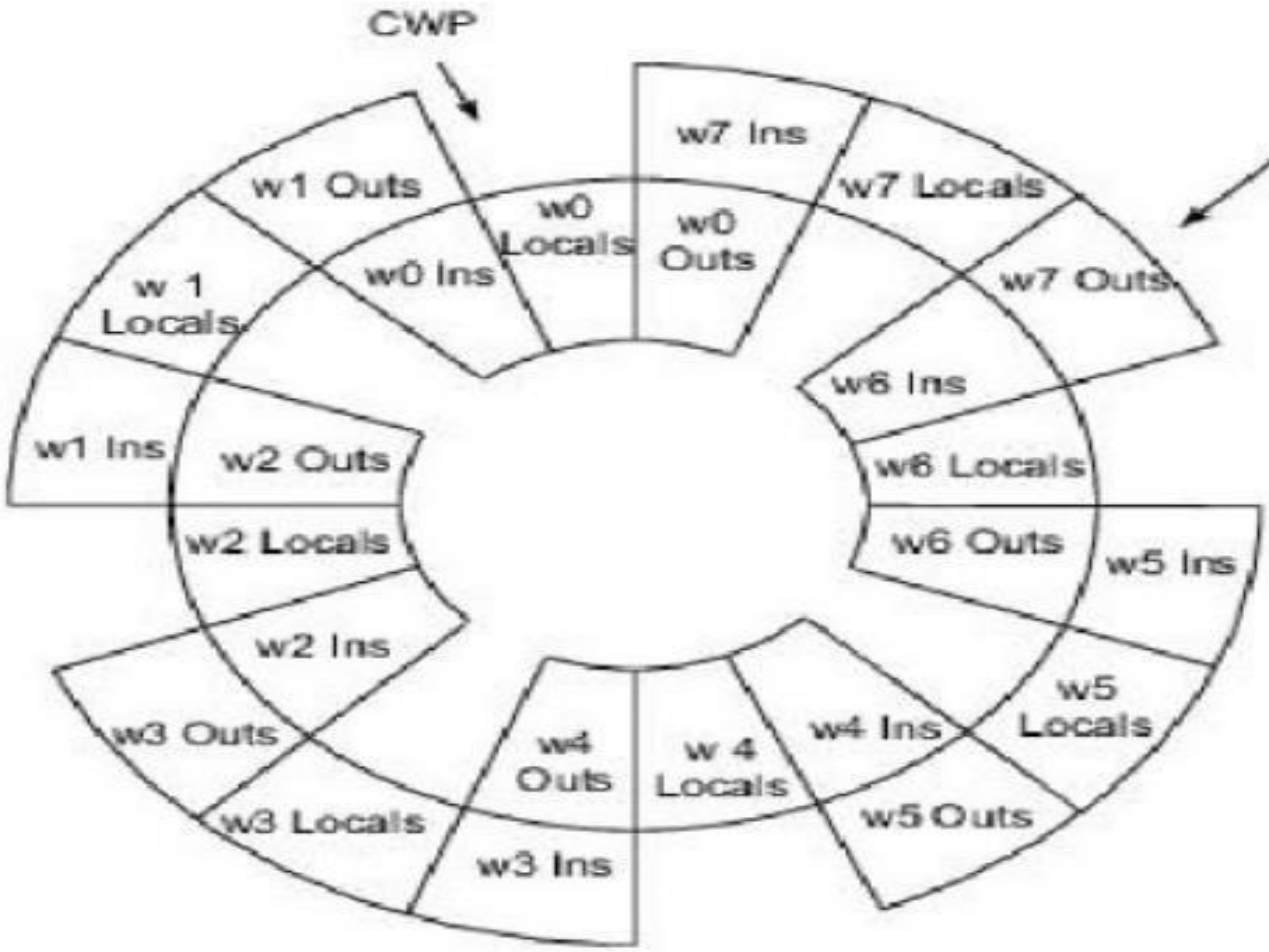
Trap base register- pointer to trap handlers

The overlapping windows save time for inter-process communication and context switching



(a) Three overlapping register windows and the globals registers

https://hemanthrajhemu.github.io



(b) Eight register windows forming a circular stack

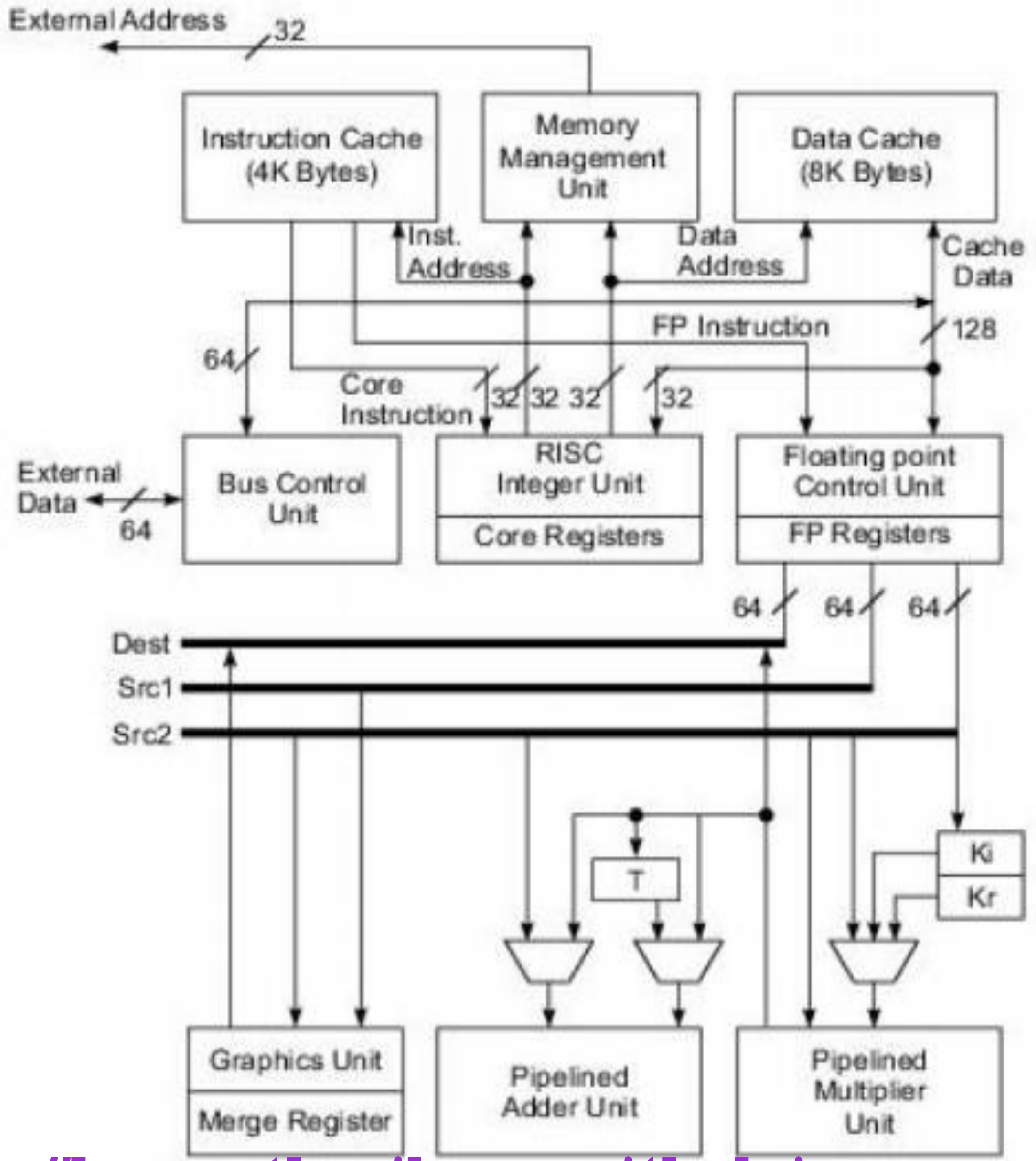


Fig. 4.9 Functional units and data paths of the Intel i860 RISC microprocessor (Courtesy of Intel Corporation, 1990)

Example 2 : Intel i860:

- 64 bit RISC processor on a single chip with > 1 million transistors
- There are nine functional units connected by multiple data paths ranging from 32 to 128 bits.
- There are two floating point units namely *multiplier unit* and *adder unit*, both of which can execute concurrently
- Address busses were 32-bit and data busses were 64- bit
- instruction cache had 4K organized as 2 way set associative with 32 bytes per block
- Data cache had 8k bytes organized as 2 way set associative, with write back policy used.
- Both integer and floating point could execute concurrently.
- Graphic unit supported 3-D drawings with color intensities.
- Merge registers was used to accumulate results
- i860 executed 82 instructions in one clock cycle (25ns)
  - 42 RISC integer
  - 24 floating point
  - 10 graphics
  - 6 assembler operations

Processors and Memory Hierarchy

- Advanced Processor Technology
    - Design Space of Processors
    - Instruction-Set Architectures
    - CISC Scalar Processors
    - RISC Scalar Processors
- Superscalar and Vector Processors
    - Superscalar Processors
    - VLIW Architecture
    - Vector and Symbolic Processors

https://hemanthrajhemu.github.io

"Scalar" vs "Superscalar" Processors

- **Scalar processors:-**
  - Execute one instruction per cycle
  - One instruction is issued per cycle
  - Pipeline throughput: one instruction per cycle
- **Superscalar processors:-**
  - Multiple instruction pipelines used
  - Multiple instruction issued per cycle and
  - Multiple results generated per cycle
  - Designed to exploit instruction-level parallelism in user programs
  - Amount of parallelism depends on the type of code being executed
  - Thus, instruction-issue degree in superscalar has been limited to 2 – 5

"Scalar" vs "Superscalar" Processors

- **Pipelining in Superscalar Processors**
  - A superscalar processor of degree m can issue m instructions per cycle
  - To fully utilize, at every cycle, there must be m instructions for execution in parallel
  - Dependence on compilers is very high
  - Figure depicts three instruction pipeline

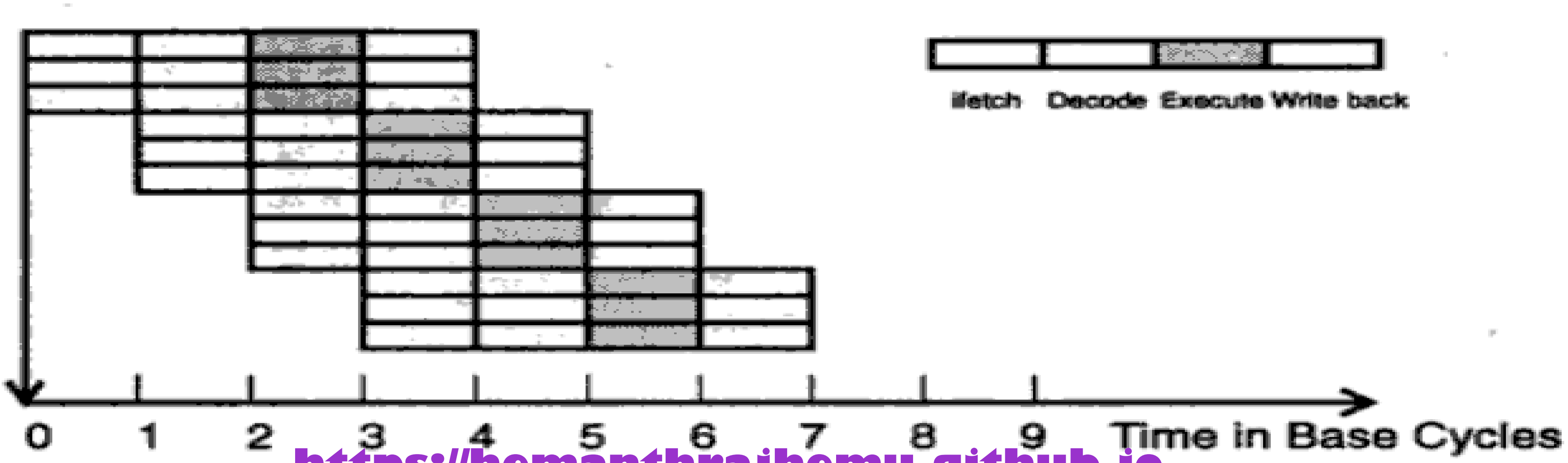


Figure 4.11 A superscalar processor of degree m = 3.

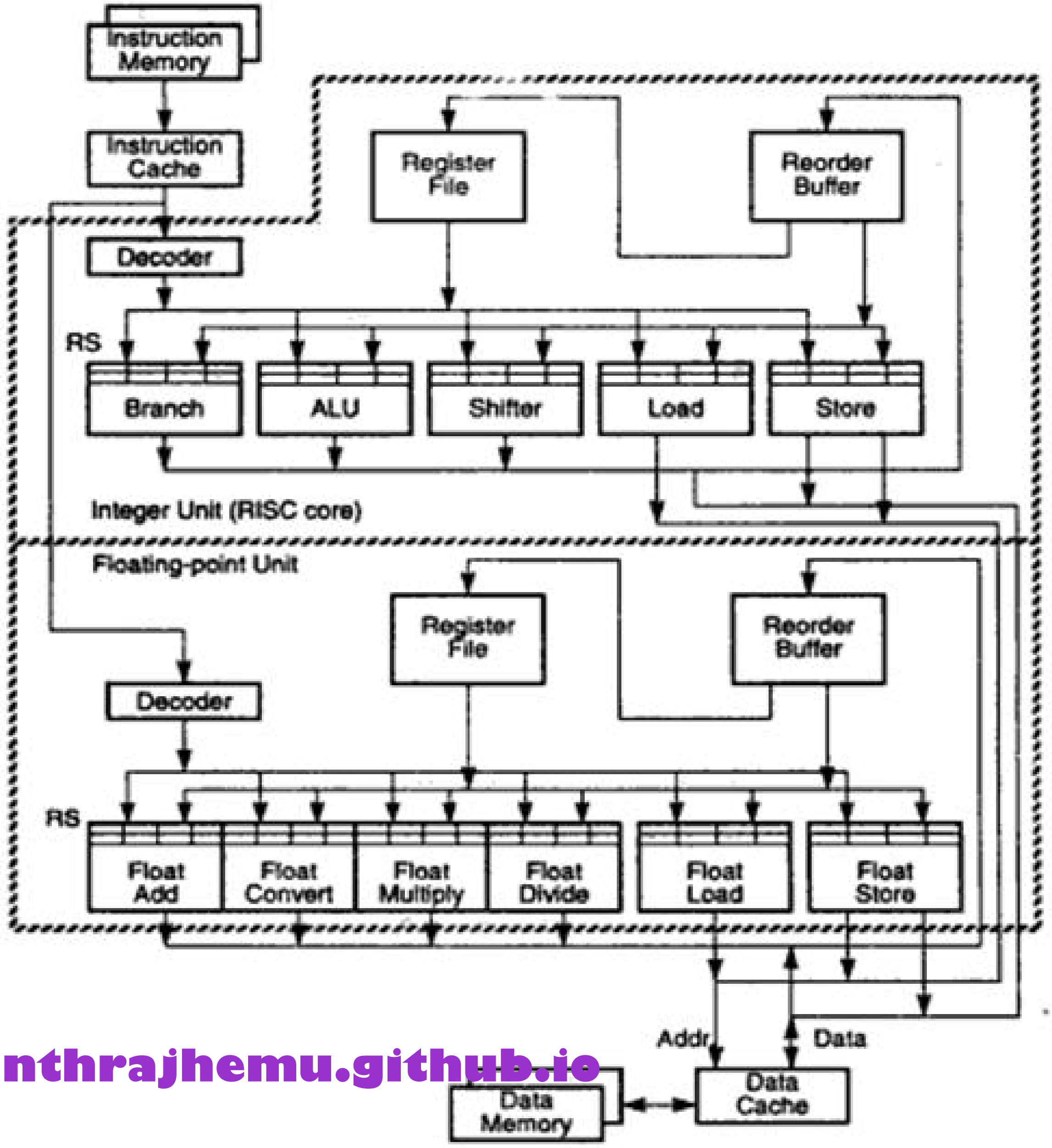
Example 1

• A typical superscalar architecture

• Multiple instruction pipelines are used, instruction cache supplies multiple instructions per fetch

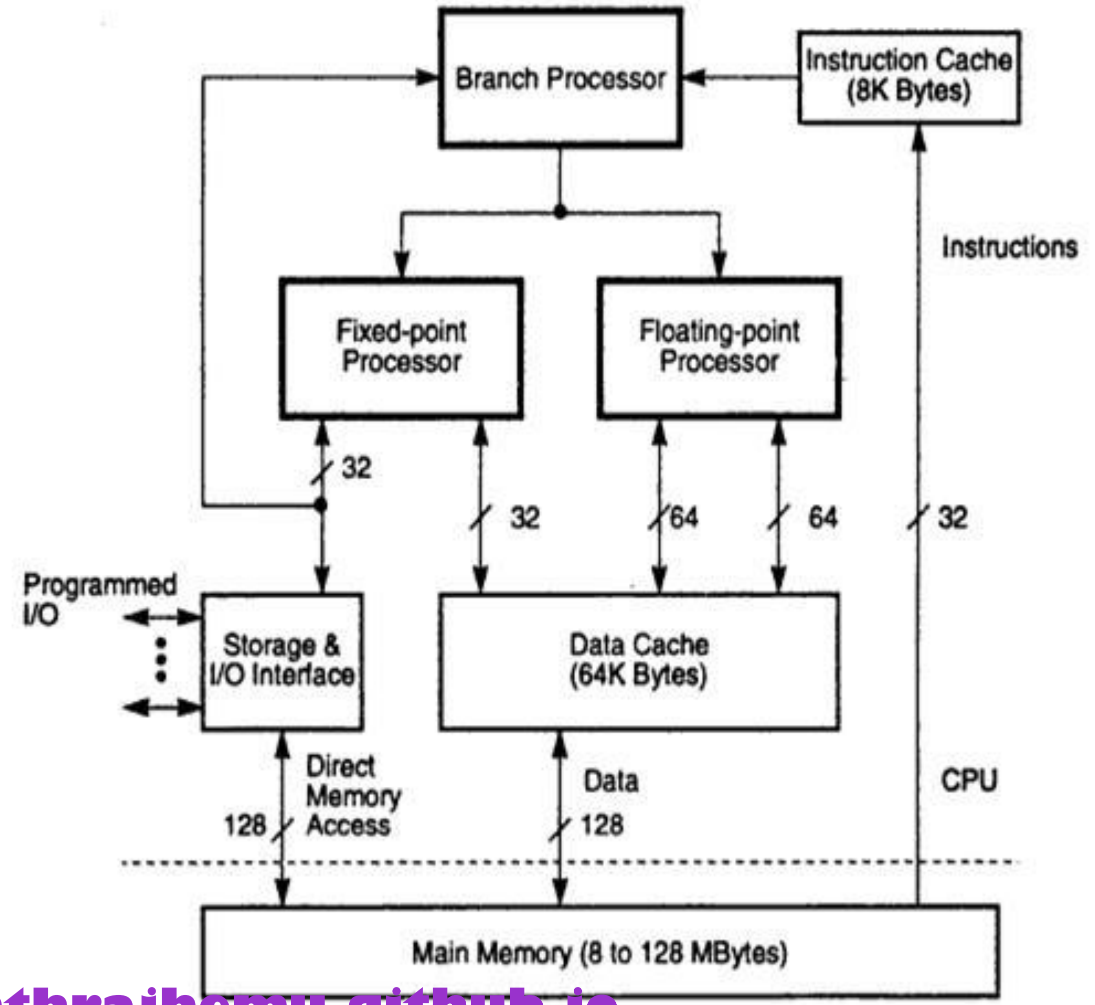
• Multiple functional units are built into *integer unit* and *floating point unit*

• Multiple data buses run though functional units, and in theory, all such units can be run simultaneously



Example 2: IBM RS/6000

- A superscalar architecture by IBM
- Three functional units namely
  - branch processor
  - fixed point processor
  - floating point processor
- At a given cycle five operations can be executed simultaneously
  - a branch
  - a condition-register operation
  - A fixed point instruction
  - a floating-point multiply-add
- Used for numeric intensive and multiuser commercial environments.



Processors and Memory Hierarchy

- Advanced Processor Technology
  - Design Space of Processors
  - Instruction-Set Architectures
  - CISC Scalar Processors
  - RISC Scalar Processors
- Superscalar and Vector Processors
  - Superscalar Processors
  - VLIW Architecture
  - Vector and Symbolic Processors

https://hemanthrajhemu.github.io

Very Large Word Instruction (VLIW) Architectures

- Typical VLIW architectures have instruction word length of hundreds of bits.
- Built upon two concepts, namely

  1. Superscalar processing

  - Multiple functional units work concurrently
  - Common large register file is shared

  2. Horizontal micro coding

  - Different fields of the long instruction word carries opcodes to be dispatched to multiple functional units
  - Programs written in conventional short opcodes are to be converted into VLIW format by compilers

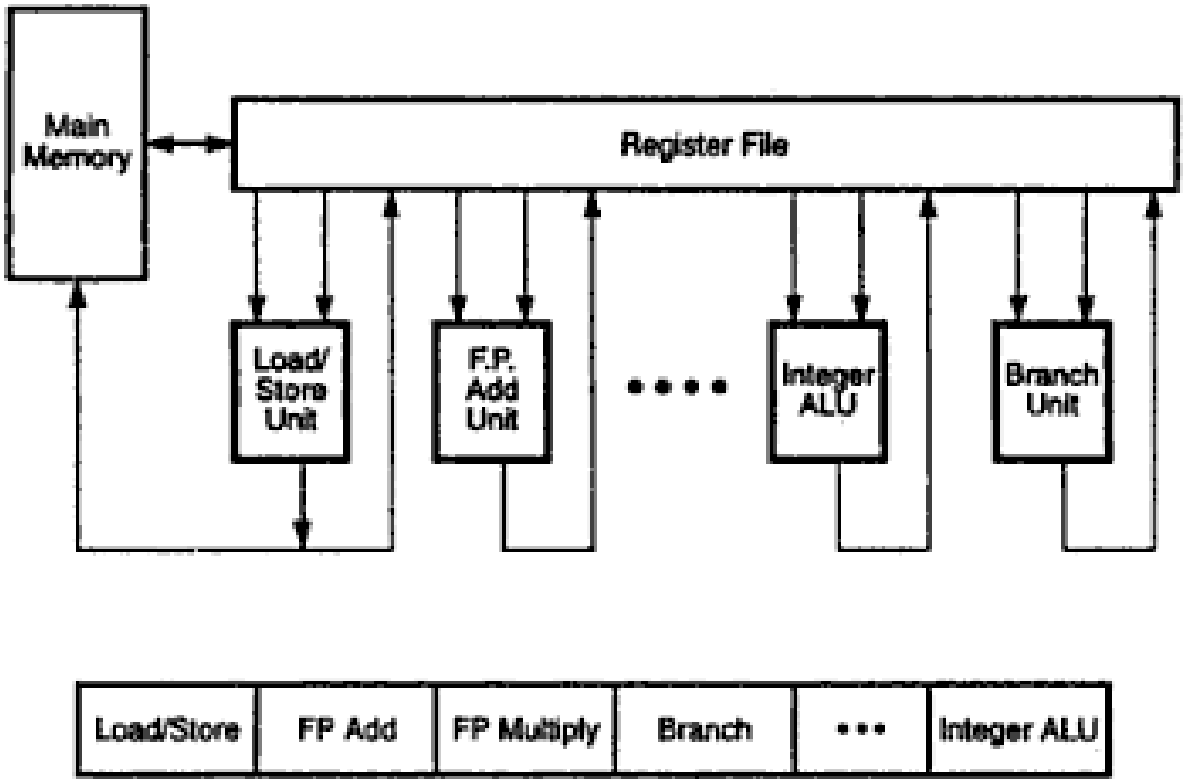
Very Large Word Instruction (VLIW) Architectures

Typical VLIW Architecture

• Multiple functional units are concurrently used

• All functional units use the same *register file*

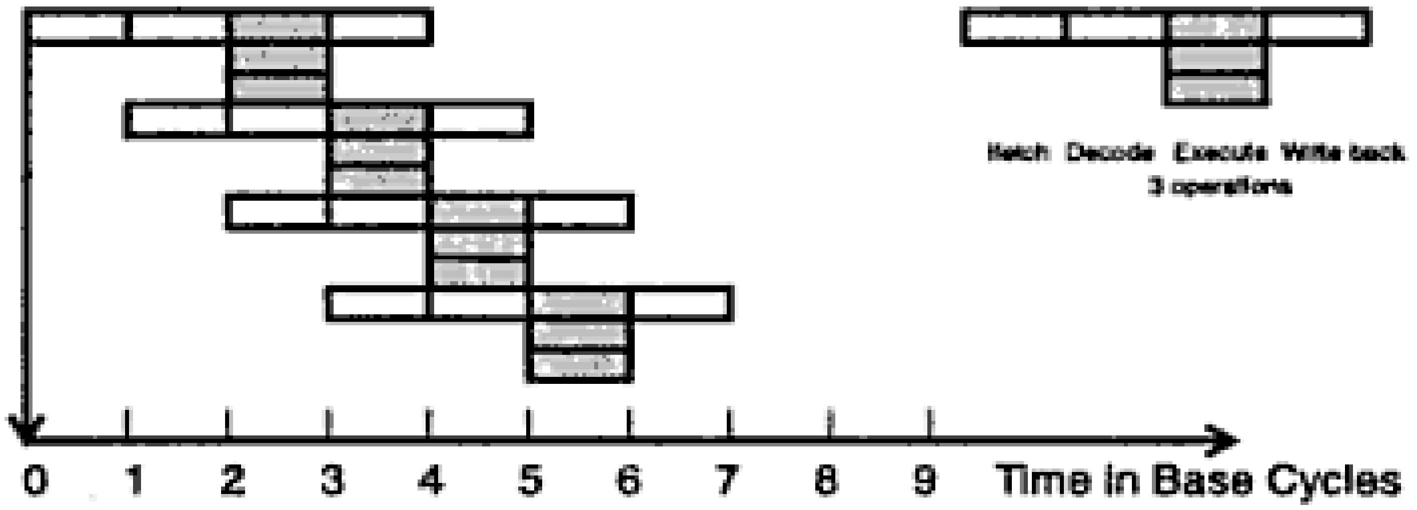
* Different fields of instructions carry opcodes to different functional unit.



(a) A typical VLIW processor and instruction format

Very Large Word Instruction (VLIW) Architectures

Pipelining in VLIW Architecture


Fetch Decode Execute Write back
3 operations
0 1 2 3 4 5 6 7 8 9 Time in Base Cycles

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

- Each instruction in VLIW architecture specifies multiple operations.
- Execute stage has multiple operations
- Instruction parallelism and data movement in VLIW architecture are specified at compile time
- CPI of VLIW architecture is lower than superscalar processor

Processors and Memory Hierarchy

- Advanced Processor Technology
  - Design Space of Processors
  - Instruction-Set Architectures
  - CISC Scalar Processors
  - RISC Scalar Processors
- Superscalar and Vector Processors
  - Superscalar Processors
  - VLIW Architecture
  - Vector and Symbolic Processors

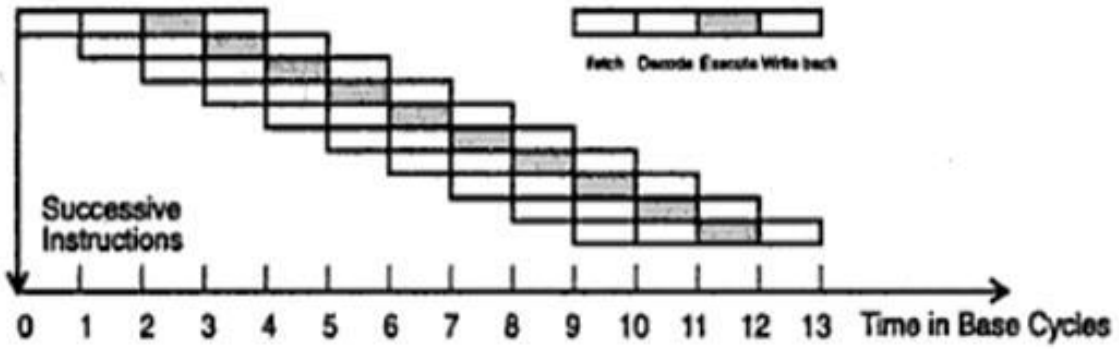
https://hemanthrajhemu.github.io

Vector Processors

- Vector processor is a coprocessor designed to perform vector computations
- Vector computations involve instructions with large array of operands
  - Same operation is performed over an array of operands
- Vector processor may be designed with :
  - Register to register architecture
    - Involves vector register files
  - Memory to memory architecture
    - Involves memory addresses

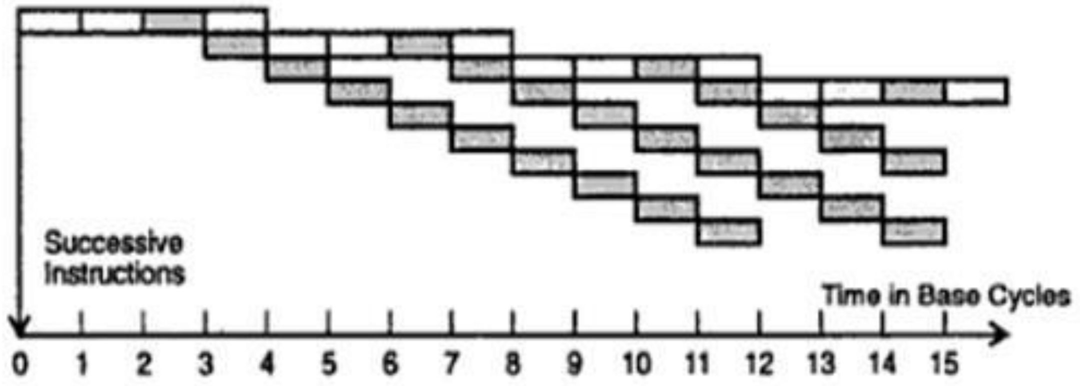
Vector Processors

- Scalar pipeline
- Each "Execute-Stage" operates upon a scalar operand


Fetch Decode Execute Write back
Successive Instructions
0 1 2 3 4 5 6 7 8 9 10 11 12 13 Time in Base Cycles

(a) Scalar pipeline execution (Fig. 4.2a redrawn)

- Vector pipeline
- Each "Execute-Stage" operates upon a vector operand

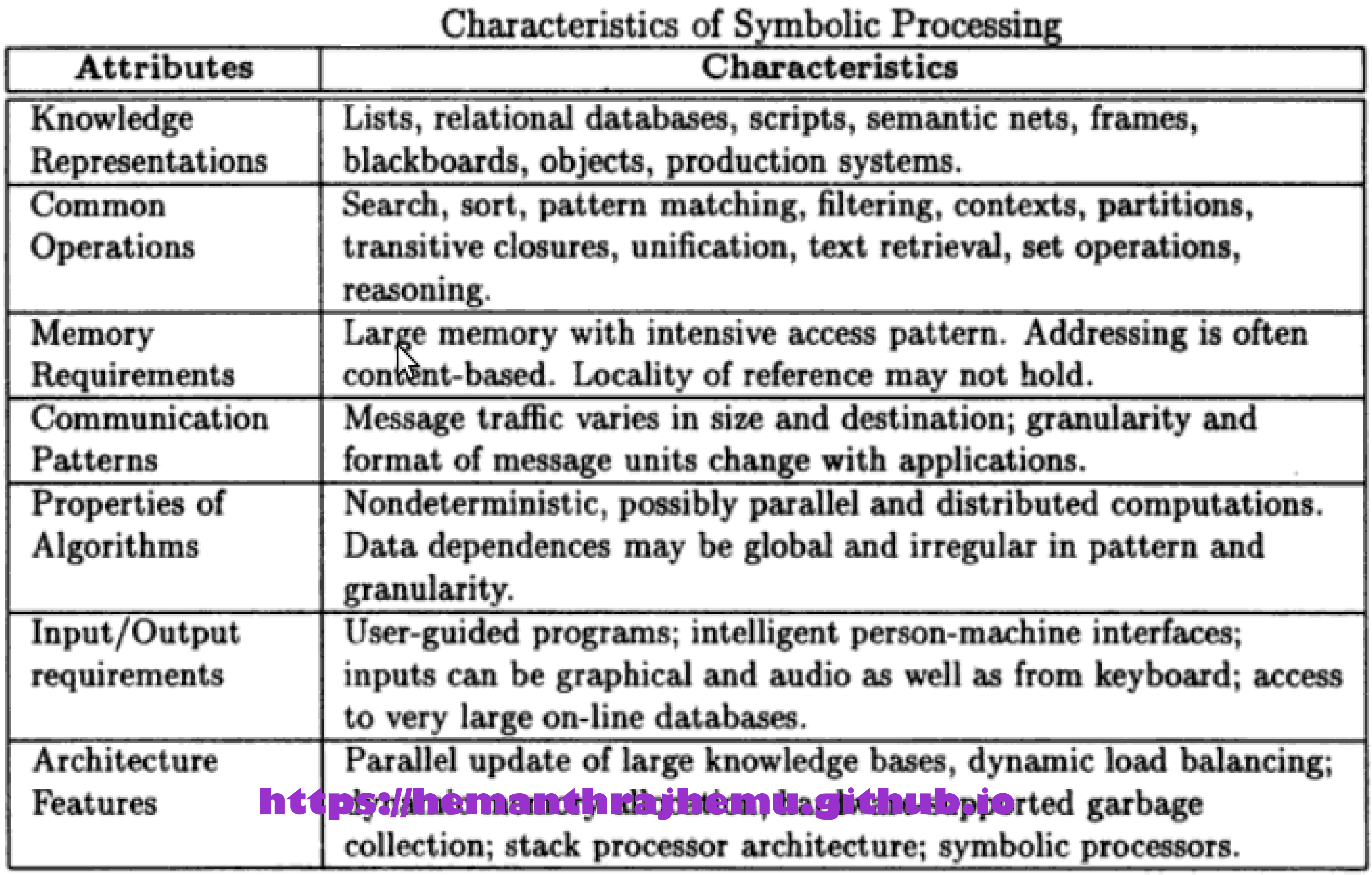


(b) Vector pipeline execution

Symbolic Processors

- Applications in the areas of pattern recognition, expert systems, artificial intelligence, cognitive science, machine learning, etc.
- Symbolic processors differ from numeric processors in terms of:-
  - Data and knowledge representations
  - Primitive operations
  - Algorithmic behavior
  - Memory
  - I/O communication

Symbolic Processors: Characteristics

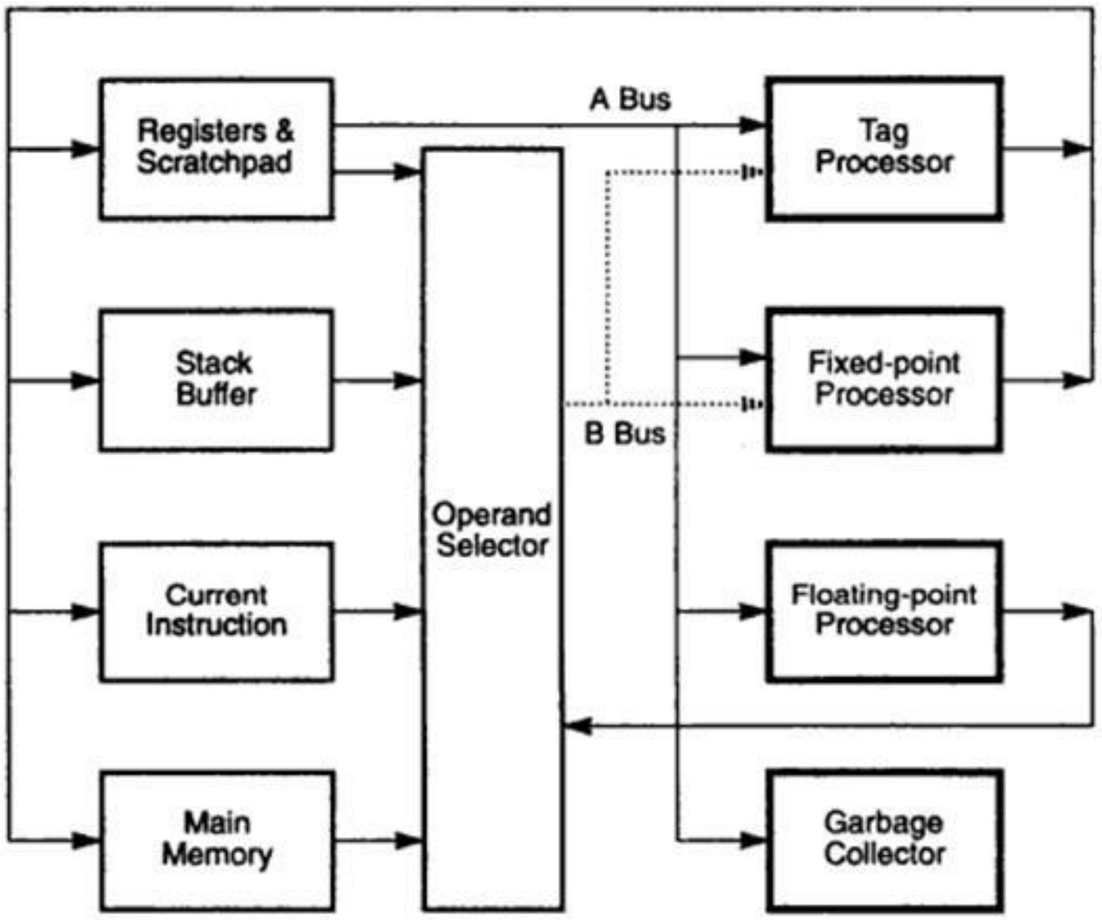


Characteristics of Symbolic Processing

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.

Example: Symbolics 3600 LISP processor

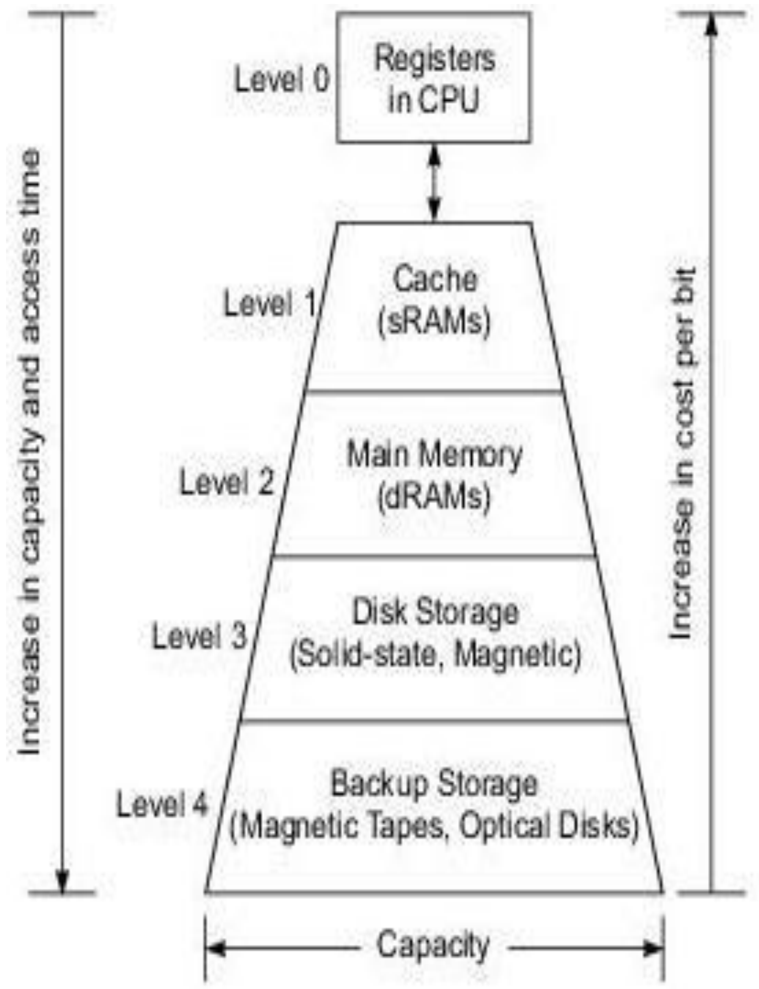
- Symbolic Lisp Processor
- Multiple processing units are provided which can work in parallel
- Operands are fetched from scratch pad or stack
- Processor executes most of the instructions in single machine cycle



Hierarchical Memory Technology

The storage at each level is categorized by five parameters

- Access time (t_i) – roundtrip time from cpu to memory
- Memory size (si)- number of bytes or words in level
- Cost per byte (ci)- product of $c_{i.}s_i$
- Transfer bandwidth (bi)- rate at which information is transferred
- Unit of transfer (xi)- grain size of data transfer


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

https://hemanthrajhemu.github.io

Hierarchical Memory Technology

1. Register and cache

- The registers are parts of the processor: on the chip or off-chip.
- Registers are operated directly by processor and operate at speed of processor.
- The cache is controlled by the MMU and is programmer-transparent
- Implemented at one or multiple levels, depending on the speed and application requirements

Hierarchical Memory Technology

2. Main memory:

- The main memory is sometimes called the primary memory of a computer system.
- Often implemented by the most cost-effective RAM chips (DDR SDRAMS)
- The main memory is managed by a MMU in cooperation with the operating system.

Hierarchical Memory Technology

3. Disk drives and backup storage:

- It holds the system programs such as the OS and compilers, user programs and data sets
- Optical disks and magnetic tape units are off-line memory for use as backup storage
- Disk drives are also available in the form of RAID (Redundant Array of Inexpensive Discs)

Hierarchical Memory Technology

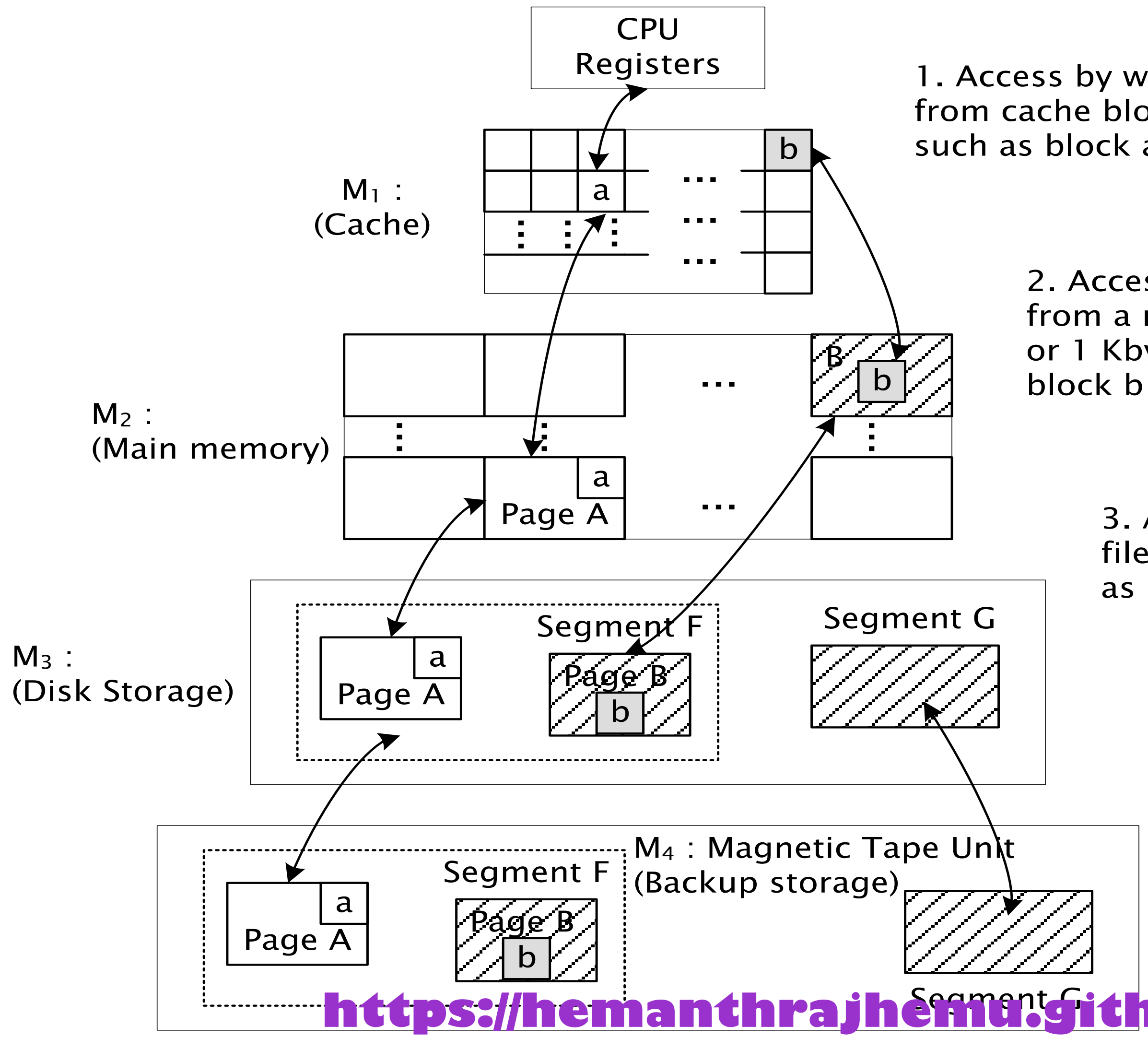
4. Peripheral Technology:

- Besides disk drives and backup storage, peripheral devices include printers, plotters, terminals, monitors, graphics displays, optical scanners, image digitizers, output microfilm devices, etc.
- The technology of peripheral devices has improved rapidly in recent years.

Table 4.7 *Memory Characteristics of a Typical Mainframe Computer in 1993*

Memory level Characteristics	*Level 0 CPU Registers*	*Level 1 Cache*	*Leve 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

Inclusion, coherence and locality



1. Access by word (4 Bytes) from cache block of 32 bytes, such as block a

2. Access by block (32 Bytes) from a memory page of 32 block or 1 Kbytes, such as block b from page B

3. Access by page (1 Kbytes) from a file consisting of many pages, such as page A and page B in segment F

4. Segment transfer with different number of pages

Inclusion, coherence and locality

- ***Inclusion Property*** : $M_n \supset M_{n-1} \dots \supset M_2 \supset M_1$
  - All information items are originally stored in outmost level M_n
  - During the processing, subset of M_n is copied into M_{n-1}, and M_{n-1} into M_{n-2},.
  - Information in M_i can be also found in M_{i+1}, M_{i+2}, …
  - Word miss in M_i implies miss in M_{i-1}, M_{i-2}, …

Inclusion, coherence and locality

- *Coherence Property*
  - Copies of the same information item at successive memory levels be consistent
  - If a word is modified in the cache, copies of that word must be updated at all higher levels
  - Frequently used information is often found in the lower levels in order to minimize the effective access time of the memory hierarchy
  - Two strategies for maintaining the coherence in a memory hierarchy
    - Write-Through (WT) - demands immediate update in Mi+1 if a word is modified in Mi for i= 1,2, , n— 1.
    - Write Back (WB) - delays the update in Mi+1 until the word being modified in Mi is replaced or removed from Mi.

Inclusion, coherence and locality

Locality of Reference: (90-10 rule)

- There are three dimensions of the locality property:
  - *Temporal Locality*
    - Referenced items are likely to be referenced again in the near future.
    - Often caused by special program constructs such as iterative loops, temporary variables, or subroutines
    - The temporal locality determines the size of memory at successive levels.

Inclusion, coherence and locality

Locality of Reference: (90-10 rule)

- There are three dimensions of the locality property:
  - *Spatial Locality*
    - This refers to the tendency for a process to access items whose addresses are near one another.
    - operations on tables or arrays involve accesses of a certain clustered area in the address space.
    - Program segments, such as routines and macros, tend to be stored in the same neighborhood of the memory space.
    - The spatial locality assists in determining the size of unit data transfers between adjacent memory levels.

Inclusion, coherence and locality

Locality of Reference: (90-10 rule)

- There are three dimensions of the locality property:
  - *Sequential Locality*
    - Typical programs, execution follows a sequential order unless branch instructions create out-of-order executions.
    - The ratio of in-order execution to out-of-order execution is roughly 5 to 1 in ordinary programs.
    - Besides, the access of a large data array also follows a sequential order.

Memory capacity planning

- The performance of a memory hierarchy is determined by the effective access time t_{eff} to any level in the hierarchy.
- lt depends on the hit ratios and access frequencies at successive levels.
- ***Hit ratio:***
  - When an information item is found in M_p we call it a hit, otherwise miss.
  - The hit ratio h_i at M_i is the probability that an information item will be found at M_i
  - The miss ratio at M_i, is defined as $1 - h_i$
  - Access frequency $f_i = (1-h_1)(1-h_2)\ldots(1-h_{i-1})h_i$
  - This implies that the inner levels of memory are accessed more often than the outer levels.

Memory capacity planning

- ***Effective access time:***
  - Every time a miss occurs, a penalty must be paid to access the next higher level of memory. (block miss/ page fault)
  - Effective memory access time

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

$$= h_1 t_1 + (1 - h_1) h_2 t_2 + (1 - h_1)(1 - h_2) h_3 t_3 + \ldots +$$

$$(1 - h_1)(1 - h_2) \ldots (1 - h_{n-1}) t_n$$

  - Still, the effective access time depends on the program behavior and memory design choices.

- ***Memory optimization:***

$$C_{total} = \sum_{i=1}^{n} c_i \cdot s_i$$

  - Therefore, the above optimization involves tradeoff among cost C_i , Size S_i, time t_i for all levels.

Virtual Memory Technology

Virtual Memory Models

- The idea is to expand the use of the physical memory among many programs with the help of an auxiliary [backup] memory such as disk arrays.
- Only active programs or portions of them become residents of the physical memory at one time.
- Active portions of programs can be loaded in and out from disk to physical memory dynamically under the coordination of the operating system.

Virtual Memory Technology

Virtual Memory Models

- *Address Space:*
  - Each word in the physical memory is identified by a unique physical Address (physical Address space)
  - Virtual Addresses are those used by machine instructions making up an executable program.
  - The virtual addresses must be translated into physical addresses at run time.
  - A system of translation tables and mapping functions are used in this process.
  - Only the active portions of running programs are brought into the main memory.

Virtual Memory Technology

- *Address Mapping*
  - Let V be the set of virtual addresses generated by a program running on a processor.
  - Let M be the set of physical addresses allocated to run this program
  - Mapping function $f_t : V \to M \cup \{\phi\}$
    - This varies time to time (coz physical memory is allocated and deallocated dynamically)

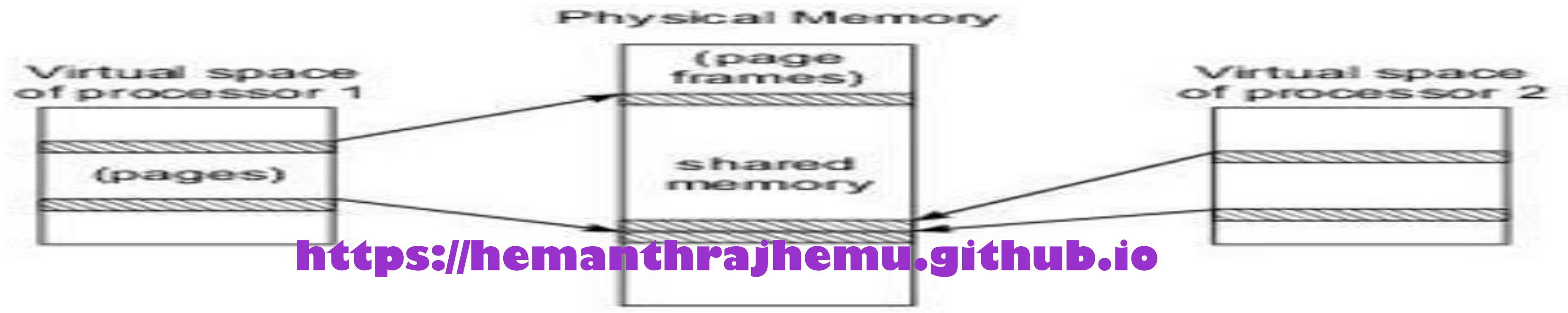
$$f_t(v) = \begin{cases} m, & \text{if } m \in M \text{ has been allocated to store the} \\ & \text{data identified by virtual address } v \\ \phi, & \text{if data } v \text{ is missing in } M \end{cases}$$

Virtual Memory Technology

- ***Address Mapping***
  - Both the virtual memory and physical memory are partitioned into fixed-length pages
  - The purpose of memory allocation is to allocate pages of virtual memory to the page frames of the physical memory.
- ***Two virtual memory models:***
  - Private virtual memory
  - Shared virtual memory

Virtual Memory Technology

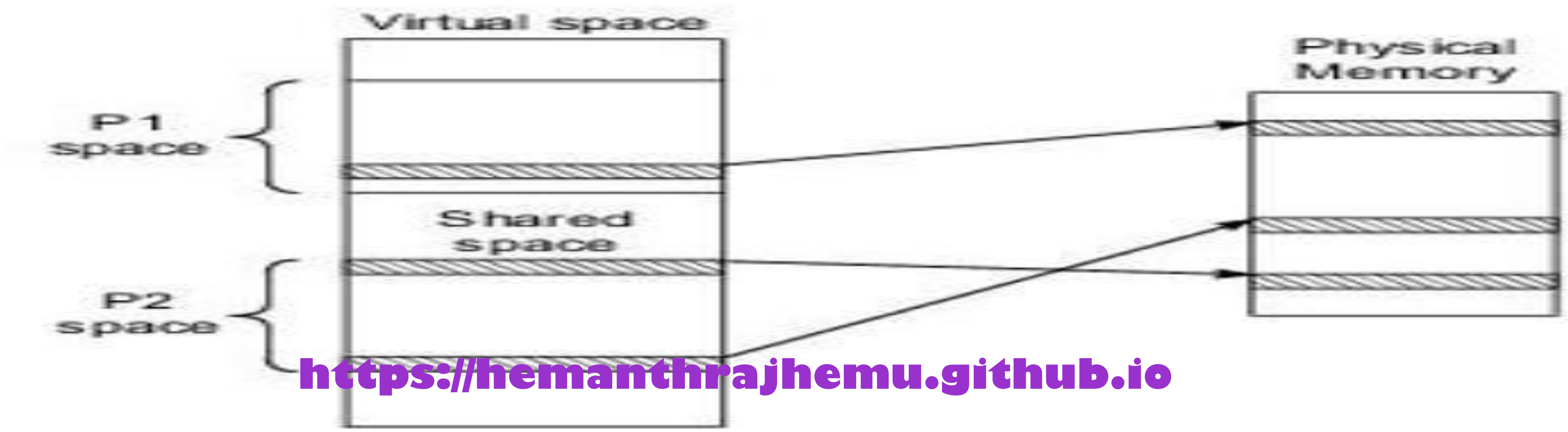
- *Private virtual memory*
  - Uses a private virtual memory address space associated with each processor.
  - Each private virtual space is divided into pages.
  - Virtual pages from different virtual spaces are mapped into the same physical memory shared by all processors.
  - The advantages of using private virtual memory include need of small processor address space [32 bits] and protection on each page.
  - No locking required.



(a) Private virtual memory space in different processors

Virtual Memory Technology

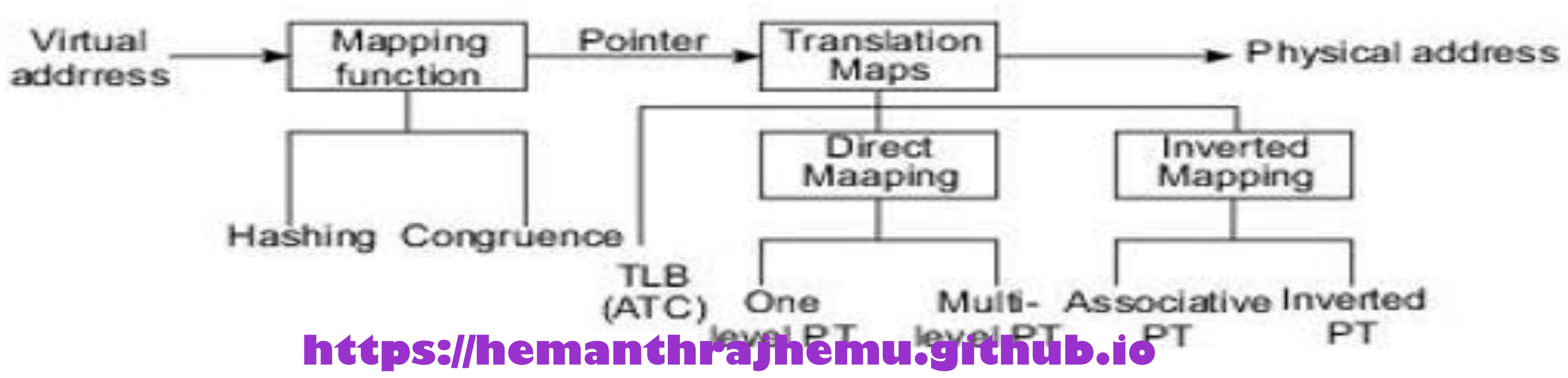
- ***Shared virtual memory***
  - This model combines all the virtual address spaces into a single globally starred virtual space
  - Each processor is given a portion of the shared virtual memory to declare their addresses
  - The advantages in using shared virtual memory is all addresses are unique. However, with large addresses.
  - Mutual exclusion {locking} is needed to enforce protected access



(b) Globally shared virtual memory space

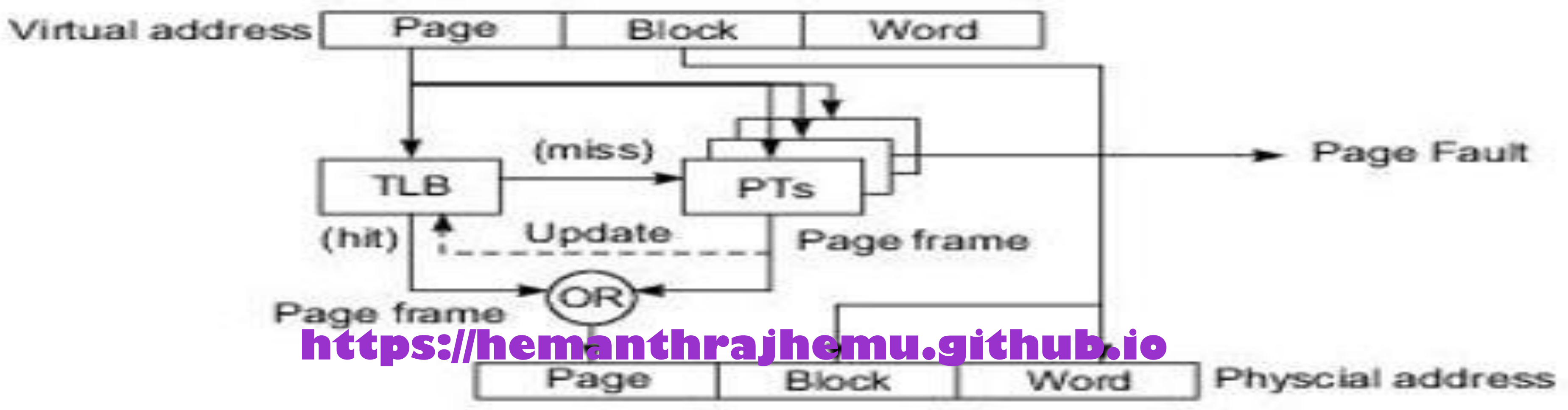
Virtual Memory Technology
TLB, Paging and Segmentation

- Address Translation Mechanism:
  - The process demands the translation of virtual addresses into physical addresses.
  - Translation maps are stored in the cache, in the main memory
  - To access these maps, a mapping function is applied to the virtual address.



(a) Virtual addrress translation schemes (PT = page table)

- Translation Look-aside Buffer
  - Translation maps appear in the form of a translation look-aside buffer (TLB) and page tables (PTs)
  - 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.
  - Each virtual address is divided into three fields:
    - The left most field holds the virtual page number,
    - The middle field identifies the each block number
    - and the rightmost field is the word address within the block.



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

Virtual Memory Technology
TLB, Paging and Segmentation

Paged memory:

- Paging is a technique for partitioning both the physical memory and virtual memory into fixed-size pages
- Page tables are used to map between pages and page frames
- If the demanded page cannot be found in the PT, a page fault is declared. (context switch)

Virtual Memory Technology
TLB, Paging and Segmentation

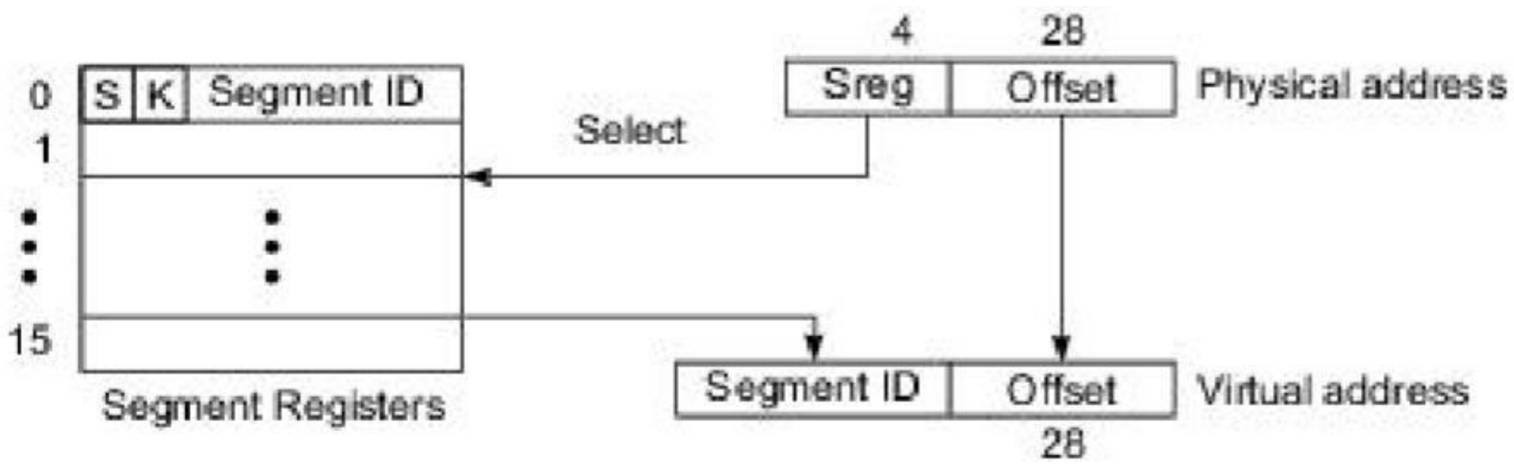
- Segmented memory
  - In a segmented memory system, user programs can be logically structured as segments.
  - Unlike pages, segments can have variable lengths.
  - The management of a segmented memory system is much more complex due to the non uniform segment size
  - Each virtual address space has a prefix field called segment number and a postfix field called the offset within the segment

Virtual Memory Technology
TLB, Paging and Segmentation

- Paged segments
  - Concepts of paging and segments combine to paged segments
  - Within each segment we have fixed size pages.
  - Each virtual address is divided into three fields.
    - Upper field – segment number
    - Middle field – page number
    - Lower field - offset within each page.

Virtual Memory Technology
TLB, Paging and Segmentation

- Inverted paging



(c) Inverted address mapping

https://hemanthrajhemu.github.io