

Computer Architecture and Organization

Module 1

Module 1 Syllabus

Module No.	Unit No.	Topics	Ref.	Hrs.
1	Title	Overview of Computer Architecture and Organization		5
	1.1	Introduction of Computer Organization and Architecture, Basic organization of computer and block level description of the functional units, Evolution of x86 Computers, Von Neumann model, Harvard Model, Embedded system	1	
	1.2	Performance Issues: Designing for performance, Amdahl's Law, Multi-core, GPGPU	1	

Introduction

- ▶ Computer has become the part and parcel of our lives
 - ▶ They are everywhere (Embedded System?)
 - ▶ Laptop, tablets, mobile phone, intelligent appliances
- ▶ It is required to know how computer works
 - ▶ What is there inside the computer?
 - ▶ How does it work?
- ▶ We distinguish between two terms: Computer Architecture and Computer Organization

Difference between Computer Architecture and Computer Organization

■ Computer Architecture

- How to integrate the component to build a computer system to achieve a desired level of performance
- Analogy of architect design for planning of building, overall layout, floorplan.

■ Computer Organization

- The actual implementation of a computer in hardware
- Civil engineers task during building construction (cement, bricks, iron rods and other building materials)

Historical Perspective

- Constant quest of building automatic computing machines have driven the development of computers
 - Initial Efforts: Mechanical devices like pulleys, levers and gears
 - During world war II: Mechanical relays to carry out computations
 - Vacuum tubes developed: First electronics computer called ENIAC
 - Semiconductor transistor developed and journey of miniaturization began.

SSI → MSI → LSI → VLSI → ULSI → Billions of transistor per chip

Evolution of Computers

Pascaline (1642)



- ✎ Blaise Pascal invented the mechanical calculator called Pascaline. This calculating machine could add and subtract two numbers directly and multiply and divide by repetition.



Pascaline signed by Pascal

Difference Engine (1812)



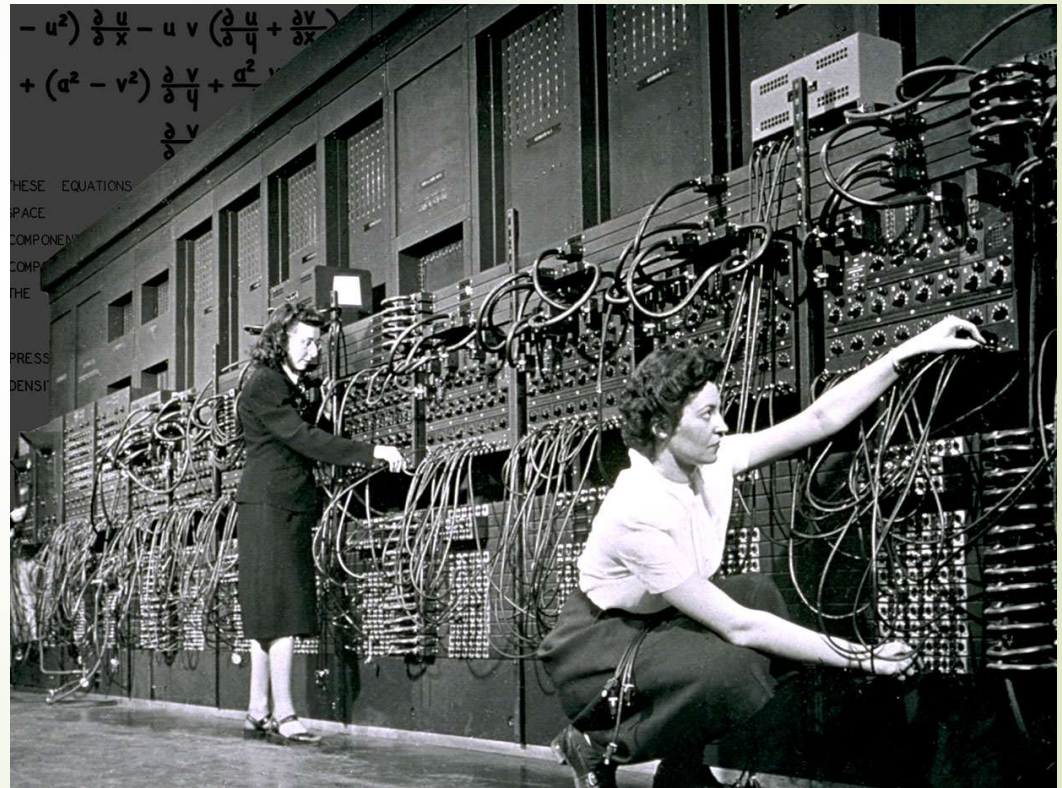
✂ Charles P. Babbage, the "father of the computer", designed a machine, the difference engine which would be steam-powered, fully automatic and commanded by a fixed instruction program.



9

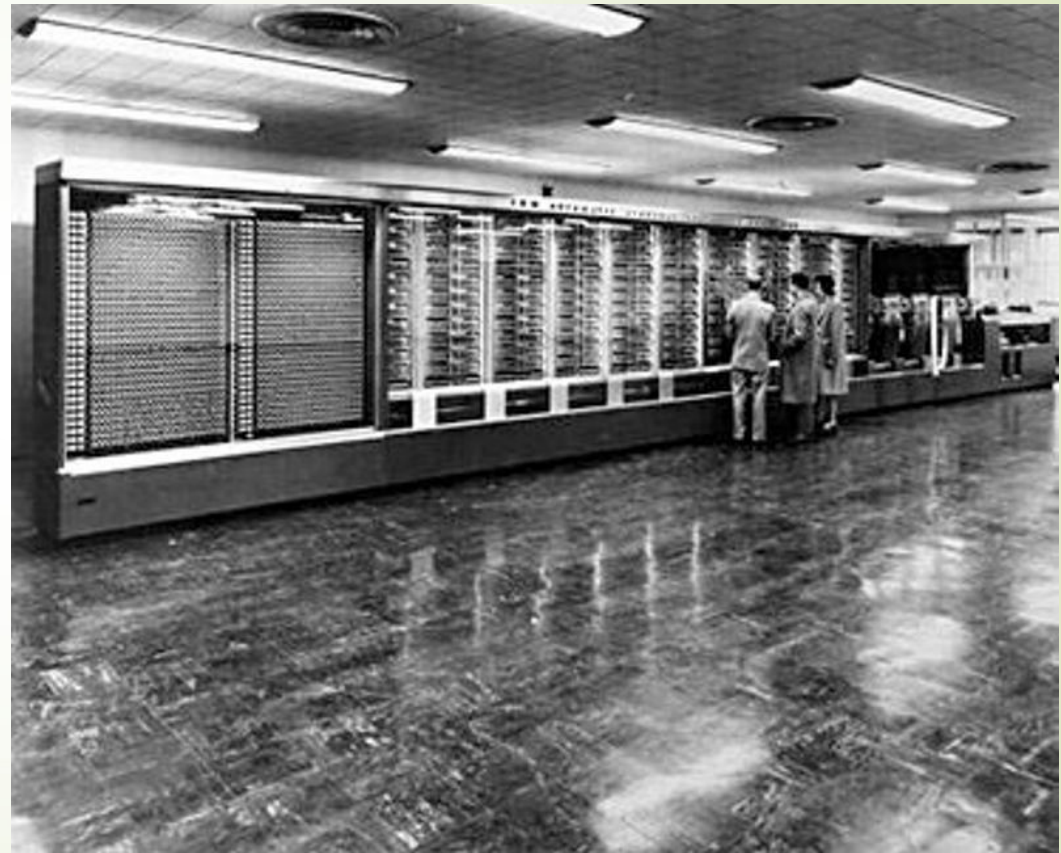
ENIAC (Electrical Numerical Integrator and Calculator)

- Developed at the University of Pennsylvania (in 1946)
- Used 18,000 Vacuum tubes, weighted 30 tubes and occupied a 30ft X 50ft space



Harvard Mark-I

- Build at University of Harvard in 1944 with support from IBM
- Use mechanical relays to represent data
- It weighted 35 tons and and required 500 miles of wiring



IBM System / 360

- Very popular mainframe computer of 60s and 70s
- Introduced many advanced architectural concepts that appeared in microprocessor several decades later



VLSI and Personal Computers



Altair 8800 - One of the first kit-based personal computers based on the Intel 8080A chip was announced in Popular Electronics and sold thousands of kits in the first month. (1975)

Generation 4 (1980) VLSI and Personal Computers



The 8086^[1] is a 16-bit microprocessor chip designed by Intel in 1978, which gave rise to the x86 architecture. Intel 8088, released in 1979, was essentially the same chip, but with an external 8-bit data bus

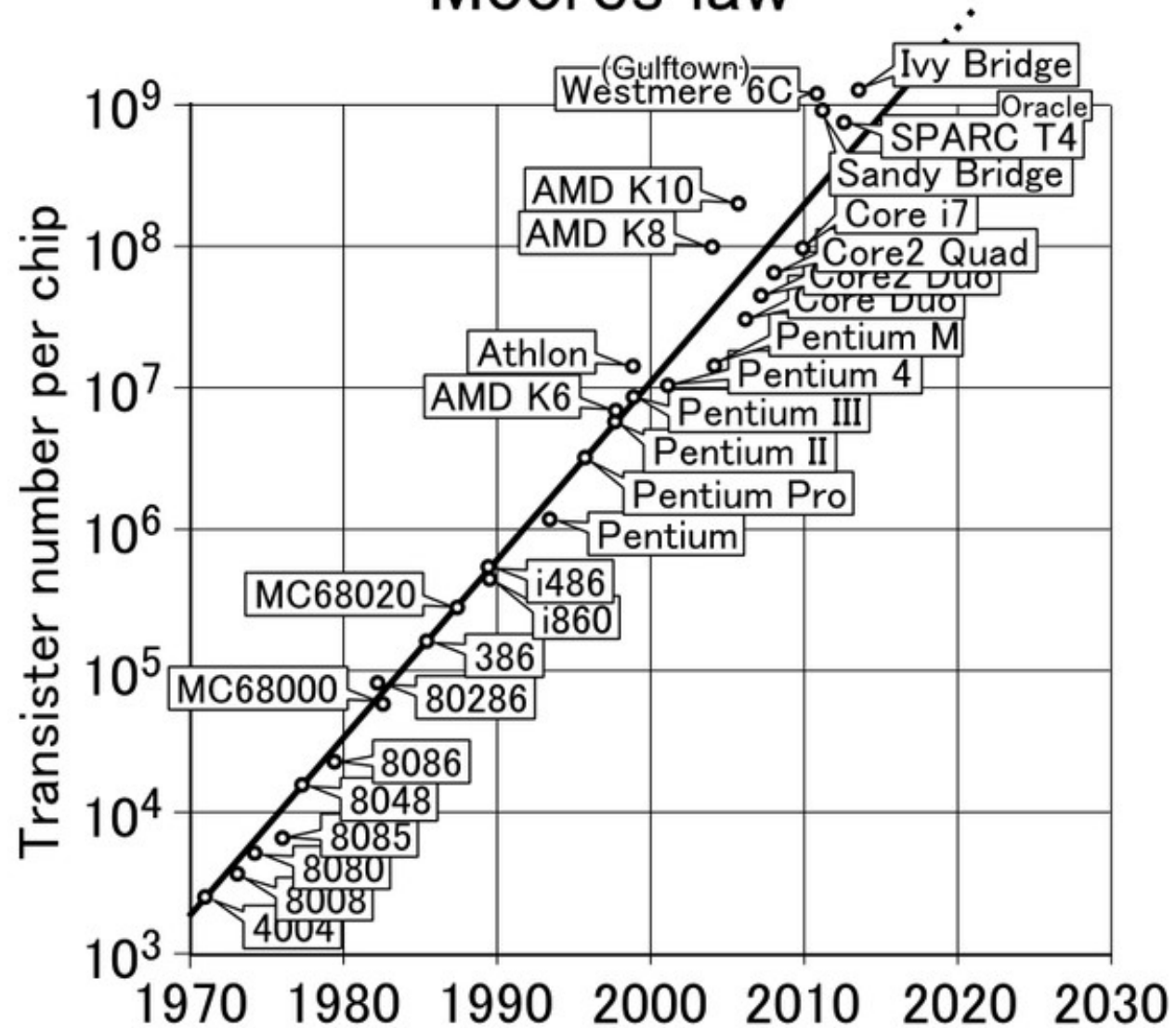
Intel Core Processors

- A modern computer chips that developed with dual core, quad core variants
- 64 bit processors that comes with various microarchitectures like Haswell, Nehlem, Sandybridge, Skylake, coffee lake etc



Generation	Main Technology	Representative Systems
First (1945-54)	Vacuum tubes, relays	Machine & assembly language ENIAC, IBM-701
Second (1955-64)	Transistors, memories, I/O processors	Batch processing systems, HLL IBM-7090
Third (1965-74)	SSI and MSI integrated circuits Microprogramming	Multiprogramming / Time sharing IBM 360, Intel 8008
Fourth (1975-84)	LSI and VLSI integrated circuits	Multiprocessors Intel 8086, 8088
Fifth (1984-90)	VLSI, multiprocessor on-chip	Parallel computing, Intel 486
Sixth (1990 onwards)	ULSI, scalable architecture, post-CMOS technologies	Massively parallel processors Pentium, SUN Ultra workstations

Moore's law



x86 Evolution

- **8080**
 - first general purpose microprocessor
 - 8 bit data path
 - Used in first personal computer – Altair
- **8086 – 5MHz – 29,000 transistors**
 - much more powerful
 - 16 bit
 - instruction cache, prefetch few instructions
- **8088 (8 bit external bus) used in first IBM PC**
- **80286**
 - addressable memory of 16 MB instead of just 1 MB and contains two modes-real mode and first generation 16-bit protected mode.
- **80386**
 - 32 bit
 - first processor to support multitasking
 - implemented the concept of paging (permitted 32-bit virtual memory address to be translated into 32-bit physical memory address)
- **80486**
 - powerful cache and instruction pipelining
 - built in maths co-processor

x86 Evolution

➤ Pentium

- use of superscalar techniques was introduced as multiple instructions started executing in parallel.

➤ Pentium Pro

- level 2 cache
- address translation in which 32-bit virtual address is translated into 36-bit physical memory address.

➤ Pentium II

- It was able to process video, audio and graphics data efficiently by incorporating Intel MMX technology

➤ Pentium 4

- address translation that translates 48-bit virtual memory address to 48-bit physical memory address

➤ Core

- First x86 with dual core that is the implementation of 2 processors on a single chip

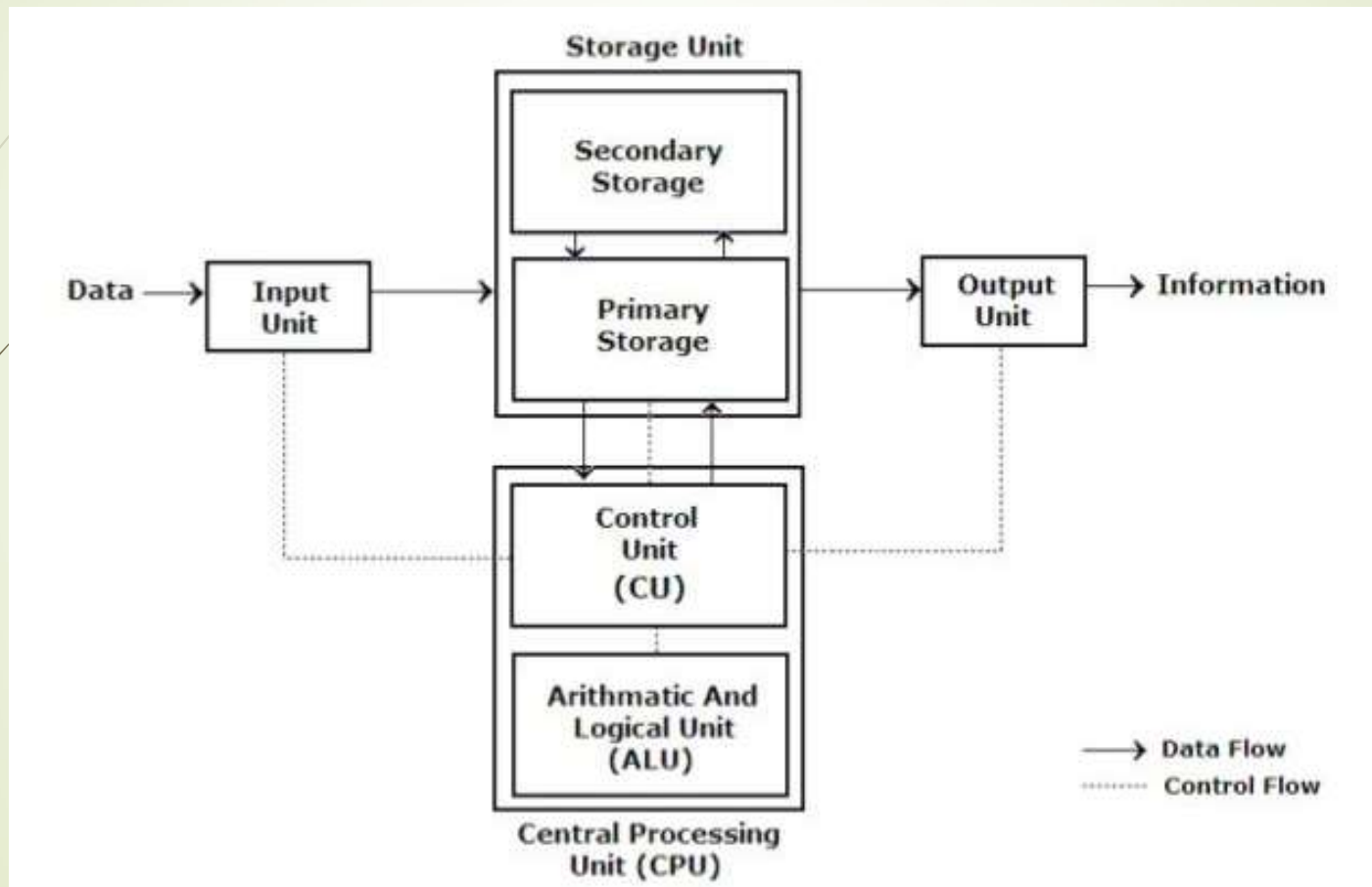
➤ Core 2

- 64 bit architecture

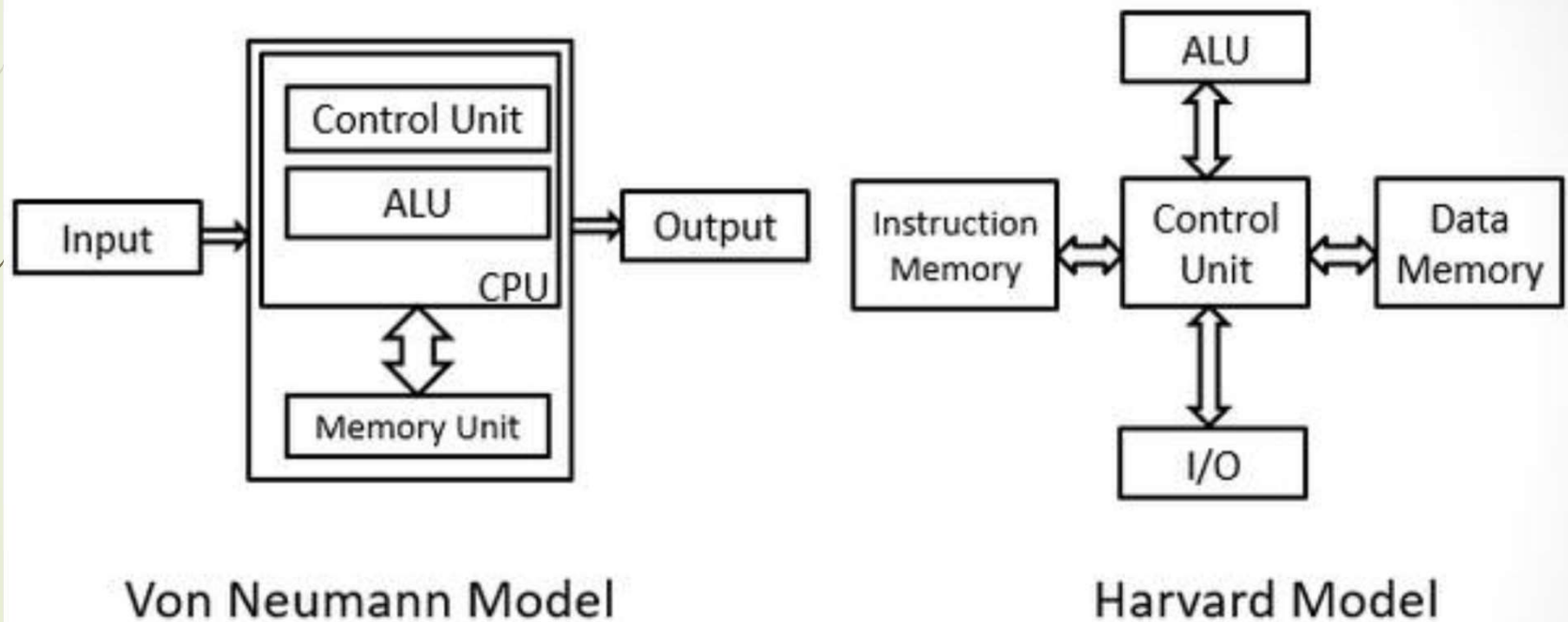
➤ Core 2 Quad – 3GHz – 820 million transistors

- Four processors on chip

Functional Block Diagram of Computer



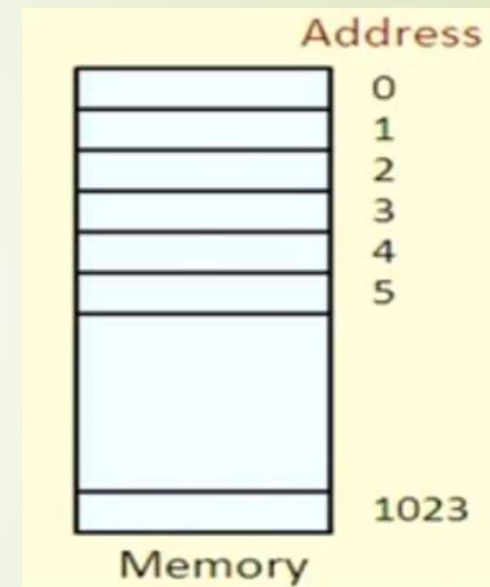
Von Neumann vs. Harvard architectures



For Interfacing with Primary Memory

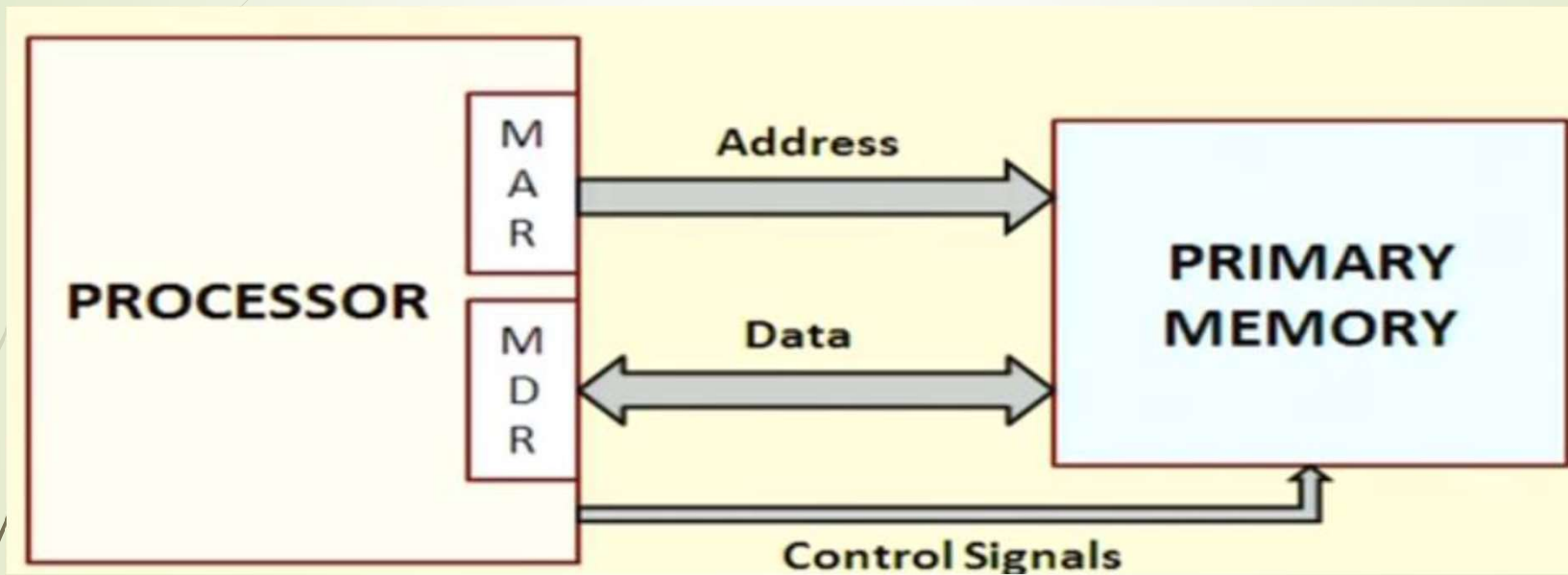
21

- Two special-purpose registers are used:
- **Memory Address register(MAR):** Holds the address of the memory location to be accessed.
- **Memory Data Register (MDR):** Holds the data that is being written into memory or will receive the data being read out from memory
- Memory considered as a linear array of storage locations (bytes or word) each with unique address.



Processor Interfacing with Primary Memory

22



Source: NPTEL Lecture of NIT Meghalaya

➡ **To read data from memory**

- a) Load the memory address into MAR**
- b) Issue the control signal *READ***
- c) *The data read from the memory is stored into MDR***

➡ **To write data into memory**

- a) Load the memory address into MAR**
- b) Load the data to be written into MDR**
- c) Issue the control signal *WRITE***

For keeping Track of Program / Instructions

Two special purpose registers are used

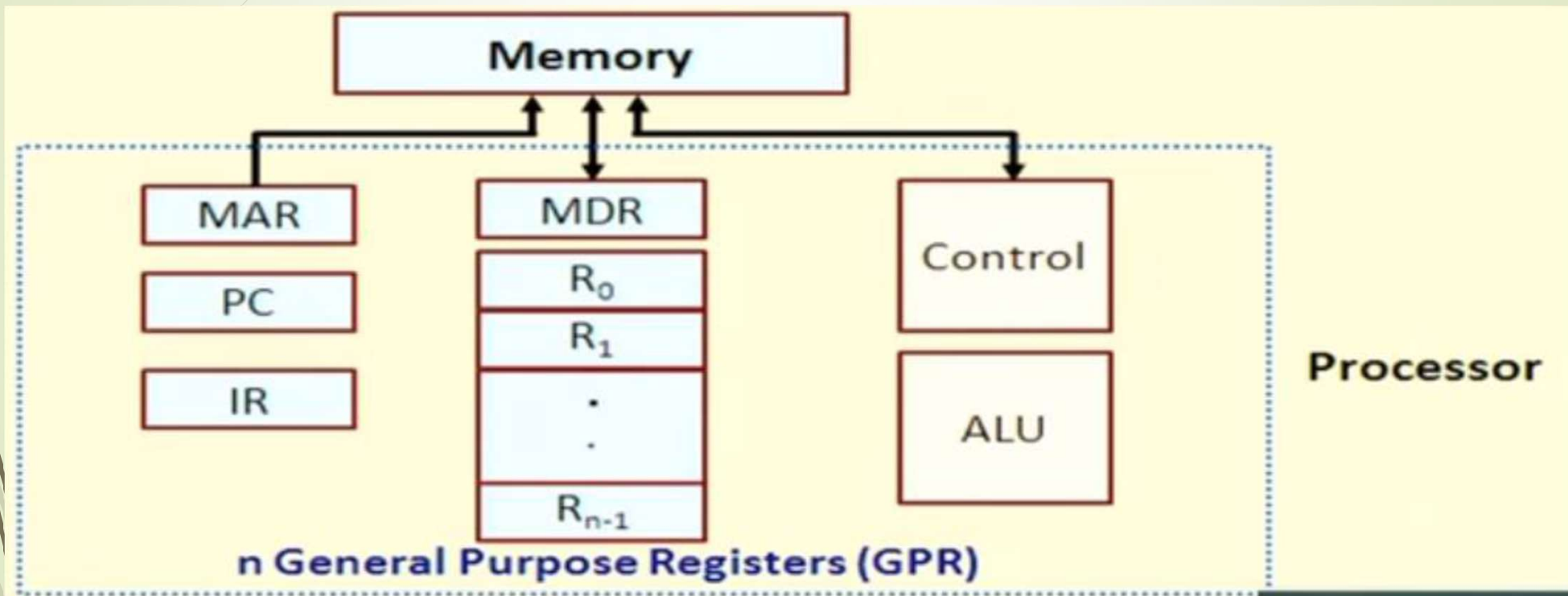
a) Program Counter (PC): Holds the memory address of the next instruction to be executed

- Automatically incremented to point to the next instruction when an instruction is being executed

b) Instruction Register (IR): Temporarily holds an instructions that has been fetched from memory

- Need to be decoded to find the instruction type
- Also contains the information about the location of data

Architecture of the Example Processor



Source: NPTEL Lecture of NIT Meghalaya

Example Instructions

- ▶ We shall illustrate the process of instruction execution with the help of the following two instructions

a) **ADD R1, LOCA**

b) **ADD R1, R2**

Execution of **ADD R1, LOCA**

- Assume that the instruction is stored in memory location 1000, the initial value of R1 is 50, and LOCA is 5000.
- Before the instruction is executed, PC contains 1000.
- Content of PC is transferred to MAR.
- READ request is issued to memory unit.
- The instruction is fetched to MDR.
- Content of MDR is transferred to IR.
- PC is incremented to point to the next instruction.
- The instruction is decoded by the control unit.

MAR \leftarrow PC

MDR \leftarrow Mem[MAR]

IR \leftarrow MDR

PC \leftarrow PC + 4



ADD R1	5000
--------	------

- LOCA (i.e. 5000) is transferred (from IR) to MAR.
- READ request is issued to memory unit.
- The data is fetched to MDR.
- The content of MDR is added to R1.

$$\text{MAR} \leftarrow \text{IR}[\text{Operand}]$$
$$\text{MDR} \leftarrow \text{Mem}[\text{MAR}]$$
$$\text{R1} \leftarrow \text{R1} + \text{MDR}$$

- LOCA (i.e. 5000) is transferred (from IR) to MAR.
- READ request is issued to memory unit.
- The data is fetched to MDR.
- The content of MDR is added to R1.

$$\text{MAR} \leftarrow \text{IR}[\text{Operand}]$$
$$\text{MDR} \leftarrow \text{Mem}[\text{MAR}]$$
$$\text{R1} \leftarrow \text{R1} + \text{MDR}$$

The steps being carried out are called micro-operations:

$$\text{MAR} \leftarrow \text{PC}$$
$$\text{MDR} \leftarrow \text{Mem}[\text{MAR}]$$
$$\text{IR} \leftarrow \text{MDR}$$
$$\text{PC} \leftarrow \text{PC} + 4$$
$$\text{MAR} \leftarrow \text{IR}[\text{Operand}]$$
$$\text{MDR} \leftarrow \text{Mem}[\text{MAR}]$$
$$\text{R1} \leftarrow \text{R1} + \text{MDR}$$

R1 50

Address	Content
1000	ADD R1, LOCA
1004	...


5000	75
------	----

LOCA

1. PC = 1000
2. MAR = 1000
3. PC = $PC + 4 = 1004$
4. MDR = ADD R1, LOCA
5. IR = ADD R1, LOCA
6. MAR = LOCA = 5000
7. MDR = 75
8. R1 = $R1 + MDR = 50 + 75 = 125$

Execution of *ADD R1,R2*

- Assume that the instruction is stored in memory location 1500, the initial value of R1 is 50, and R2 is 200.
- Before the instruction is executed, PC contains 1500.
- Content of PC is transferred to MAR.
- READ request is issued to memory unit.
- The instruction is fetched to MDR.
- Content of MDR is transferred to IR.
- PC is incremented to point to the next instruction.
- The instruction is decoded by the control unit.
- R2 is added to R1.



ADD R1, R2

MAR \leftarrow PC

MDR \leftarrow Mem[MAR]

IR \leftarrow MDR

PC \leftarrow PC + 4

R1 \leftarrow R1 + R2

R1 50

R2 200

Address	Instruction
1500	ADD R1, R2
1504	...

1. PC = 1500
2. MAR = 1500
3. PC = $PC + 4 = 1504$
4. MDR = ADD R1, R2
5. IR = ADD R1, R2
6. R1 = $R1 + R2 = 250$

Example of IAS Machine: Von Neumann Machine

- 1000 storage locations called words
- Word length – 40 bits
- A word may contain
 - A number stored as 40 binary digits (bits) – sign bit + 39 bits value
 - An instruction pair each instruction:
 - An opcode (8 bits)
 - An address (12 bits)

Structure of IAS

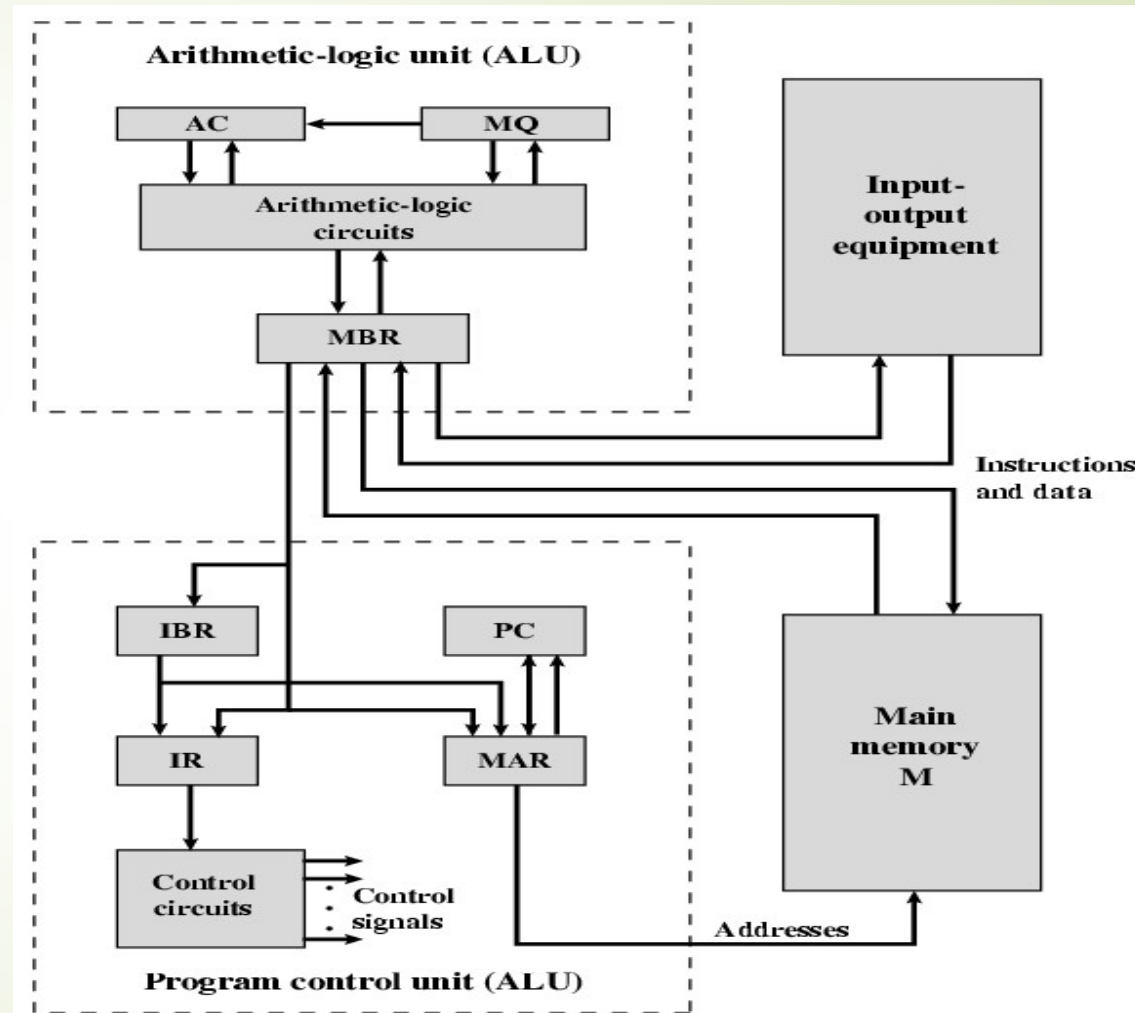
Memory

1. Load M(X) 500, ADD M(X) 501
2. STOR M(X) 500, (other instruct)
-

500. 3

501. 4

PC	
MAR	
MBR	
IR	
IBR	
AC	



Performance Issues

Introduction

- Processor execute instruction in synchronous manner using a clock that runs at a constant **clock rate** or **frequency f**
- Clock cycle time **C** is the reciprocal of the clock rate **f**
 $C = 1 / f$
- The clock rate **f** depends on two factors:
 - a) The implementation technology used
 - b) The CPU organization used
- The machine instruction typically consists of number of elementary microoperations that vary in number and complexity depending on the instructions and the CPU organization used

- A micro-operation is a elementary hardware operation that can be carried out in one cycle.
- A single machine instruction may take one or more than one CPU cycles to complete
- We can characterize the instruction by *Cycles Per Instruction (CPI)*
- Average CPI of the program
- Different instructions can have different CPIs

- For given program compiled to run on specific machine, we can define the following parameters
 - a) The total number of instructions executed or **instruction count (IC)**
 - b) The average number of **cycles per instructions (CPI)**
 - c) **Clock cycle time (C)** of the machine
- The total execution can be computed as

$$\text{Execution Time } XT = IC \times CPI \times C$$

- How do we evaluate and compare the performances of several machines?

By Measuring the Execution Times

- We measure the execution times of a program on two machines (A and B) as XT_A and XT_B .
- Performance can be defined as the reciprocal of execution time :

$$Perf_A = 1 / XT_A$$

$$Perf_B = 1 / XT_B$$

- We can estimate the speedup of machine A over machine B as:

$$Speedup = Perf_A / Perf_B = XT_B / XT_A$$

Example

- ▶ A program is run on three different machines A, B and C. Execution times are 10, 25 and 75 are noted.

Example

- ▶ A program is run on three different machines A, B and C. Execution times are 10, 25 and 75 are noted.
 - ▶ A is 2.5 times faster than B
 - ▶ A is 7.5 times faster than C
 - ▶ B is 3.0 times faster than C
- ▶ Simple for one program. But the main challenge is to extend the comparison when we have set of programs.....

Example 1

- ▶ A program is running on a machine with the following parameters
 - ▶ Total number of instructions executed = 50,000,000
 - ▶ Average CPI for the program = 2.7
 - ▶ CPU clock rate = 2.0 GHz (i.e $C = 0.5$ nsec)

Execution time of the program?

Example 1

- ▶ A program is running on a machine with the following parameters
 - ▶ Total number of instructions executed = 50,000,000
 - ▶ Average CPI for the program = 2.7
 - ▶ CPU clock rate = 2.0 GHz (i.e C = 0.5 nsec)
- ▶ Execution time of the program

$$XT = IC \times CPI \times C$$

$$XT = 50,000,000 \times 2.7 \times 0.5 \times 10^{-9} = 0.0675 \text{ sec}$$

Factors affecting Performance

	C	CPI	IC
Hardware Technology (VLSI)	X		
Hardware Technology (Organization)	X	X	
Instruction Set Architecture		X	X
Compiler Technology		X	X
Program		X	X

A tradeoff:

- **RISC** : increases number of instructions/ program, but decreases CPI and clock cycle time because the instructions and hence the implementations are simple.
- **CISC**: decreases number of instructions/ program, but increases CPI and clock cycle time because many instructions are more complex.

Example 2

- Suppose a machine A executes a program with an average CPI of 2.3. Consider another machine B (with same instruction set and a better compiler) that executes the same program with 20% less instructions and with a CPI of 1.7 at 1.2 GHz

What should be the clock rate of A so that the two machines have the same performance?

Example 2

- Suppose a machine A executes a program with an average CPI of 2.3. Consider another machine B (with same instruction set and a better compiler) that executes the same program with 20% less instructions and with a CPI of 1.7 at 1.2 GHz

What should be the clock rate of A so that the two machines have the same performance?

We must have $IC_A \times CPI_A \times C_A = IC_B \times CPI_B \times C_B$

Hence $IC_A \times 2.3 \times C_A = 0.8 \times IC_A \times 1.7 \times (1 / (1.2 \times 10^9))$

$C_A = 0.49 \times 10^{-9} \text{ sec}$

Clock rate = 2.05 GHz

Example 3

- Consider IC = 50,000,000; average CPI = 2.7 and clock rate = 2.0 GHz

Suppose we use a new compiler on the same program for which:

New IC = 40,000,000

New CPI = 3.0 (New compiler is using more complex instructions)

Also we have faster CPU implementation, with clock rate = 2.5GHz

Example 3

- Consider IC = 50,000,000; average CPI = 2.7 and clock rate = 2.0 GHz

Suppose we use a new compiler on the same program for which:

New IC = 40,000,000

New CPI = 3.0 (New compiler is using more complex instructions)

Also we have faster CPU implementation, with clock rate = 2.5GHz

$$\text{Speedup} = XT_{\text{old}} / XT_{\text{new}}$$

$$= (50,000,000 \times 2.7 \times 0.5 \times 10^{-9}) / (40,000,000 \times 3.0 \times 0.4167 \times 10^{-9})$$

$$= 1.35 \rightarrow 35 \% \text{ faster}$$

Example 4

- Consider an implementation of a ISA where the instructions can be classified into four types, with CPI values of 1, 2, 3 and 4 respectively.

Two code sequences have the following instructions counts:

Code Sequence	IC _{type1}	IC _{type2}	IC _{type3}	IC _{type4}
CS-1	20	15	5	2
CS-2	10	12	10	4

Compare CS-1 and CS-2 on basis of CPI

Example 4

- Consider an implementation of a ISA where the instructions can be classified into four types, with CPI values of 1, 2, 3 and 4 respectively.

Two code sequences have the following instructions counts:

Code Sequence	IC _{type1}	IC _{type2}	IC _{type3}	IC _{type4}
CS-1	20	15	5	2
CS-2	10	12	10	4

Compare CS-1 and CS-2 on basis of CPI

CPU cycles for CS-1: $20 \times 1 + 15 \times 2 + 5 \times 3 + 2 \times 4 = 73$

CPI for CS-1: $73 / 42 = 1.74$

CPU cycles for CS-2: $10 \times 1 + 12 \times 2 + 10 \times 3 + 4 \times 4 = 80$

CPI for CS-2: $80 / 36 = 2.22$

Instruction Frequency and CPI

- CPI can be expressed in terms of the frequencies of the various instruction types that are executed in a program.
- F_i denotes the frequency of execution of instruction type i

$$\begin{aligned} \text{CPI} &= \frac{\sum_{i=1}^n (IC_i \times CPI_i)}{IC} \\ &= \sum_{i=1}^n \left(\frac{IC_i}{IC} \times CPI_i \right) \end{aligned}$$

$$F_i = \frac{IC_i}{IC}$$

$$\text{CPI} = \sum_{i=1}^n (F_i \times CPI_i)$$

Example 5

Suppose for an implementation of a RISC ISA there are four instruction types, with their frequency of occurrence (for a typical mix of programs) and CPI as shown in the table below.

Type	Frequency	CPI
Load	20 %	4
Store	8 %	3
ALU	60 %	1
Branch	12 %	2

Find the CPI of the program

Example 5

- Suppose for an implementation of a RISC ISA there are four instruction types, with their frequency of occurrence (for a typical mix of programs) and CPI as shown in the table below

Type	Frequency	CPI
Load	20%	4
Store	8%	3
ALU	60%	1
Branch	12%	2

Find the CPI of the program,

Example 5

- Suppose for an implementation of a RISC ISA there are four instruction types, with their frequency of occurrence (for a typical mix of programs) and CPI as shown in the table below

Type	Frequency	CPI
Load	20%	4
Store	8%	3
ALU	60%	1
Branch	12%	2

Find the CPI of the program,

$$CPI = \sum_{i=1}^n (F_i \times CPI_i)$$

$$\begin{aligned} CPI &= (0.20 \times 4) + (0.08 \times 3) + (0.60 \times 1) \\ &\quad + (0.12 \times 2) \\ &= 1.88 \end{aligned}$$

Example 6

- Suppose that a program is running on a machine with the following instructions types, CPI values and the frequencies of occurrence.
- The designer gives two options: (a) Reduce CPI of instruction type A to 1.1, and (b) reduce CPI of instruction type B to 1.6. Which one is better?

Type	CPI	Frequency
A	1.3	60%
B	2.2	10%
C	2.0	30%

Example 6

- Suppose that a program is running on a machine with the following instructions types, CPI values and the frequencies of occurrence.
- The designer gives two options: (a) Reduce CPI of instruction type A to 1.1, and (b) reduce CPI of instruction type B to 1.6. Which one is better?

Type	CPI	Frequency
A	1.3	60%
B	2.2	10%
C	2.0	30%

$$\text{Average CPI for (a): } 0.60 \times 1.1 + 0.10 \times 2.2 + 0.30 \times 2.0 = 1.48$$

$$\text{Average CPI for (b): } 0.60 \times 1.3 + 0.10 \times 1.6 + 0.30 \times 2.0 = 1.54$$

Option (a) is better

Choice of Benchmark

Introduction

59

➤ Basic Concept:

- How to compare the performances of two or more computer systems?
- Set of standard programs used for comparison is called Benchmark
- Various metrics have been proposed to carry out the evaluation
 - a) MIPS (Million instructions Per Second)

a) MIPS (Million instructions Per Second)

- Computed as $(IC / XT) \times 10^{-6}$
- Dependent on instruction set
- MIPS varies between programs running on the same processor
- The MIPS rating is only valid to compare the performance of two or more processors provided that the following conditions are satisfied:
 - a) The same program is used
 - b) The same ISA is used
 - c) The same compiler is used

b)MFLOPS (Million Floating Point Operations Per Second)

- More suitable for applications that involve lot of floating-point computations
- Simply computes number of floating point operations per second
- Compilers have no floating point operations and has a MFLOP rating of 0.
- Hence not very suitable to use this metric across machines and also across programs

Example 1

- Consider a processor with three instruction classes A, B and C with the corresponding CPI values being 1, 2 and 3 respectively. The processor runs at a clock rate of 1 GHz. For a given program written in C, two compilers produce the following executed instruction counts.

	Instruction Count (in millions)		
	For IC_A	For IC_B	For IC_C
Compiler 1	7	2	1
Compiler 2	12	1	1

- Compute the MIPS rating and the CPU time for the two program versions

$$\text{MIPS} = \text{Clock Rate (MHz)} / \text{CPI}$$

$$\text{CPI} = \text{CPU Execution Cycles} / \text{Instruction Count}$$

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} / \text{Clock Rate}$$

- Solution:

- For compiler 1:

$$\text{CPI}_1 = (7 \times 1 + 2 \times 2 + 1 \times 3) / (7 + 2 + 1) = 14 / 10 = 1.40$$

$$\text{MIPS Rating}_1 = 1000 \text{ MHz} / 1.40 = 714.3 \text{ MIPS}$$

$$\text{CPU Time}_1 = ((7 + 2 + 1) \times 10^6 \times 1.40) / (1 \times 10^9) = 0.014 \text{ sec}$$

- For compiler 2:

$$\text{CPI}_2 = (12 \times 1 + 1 \times 2 + 1 \times 3) / (12 + 1 + 1) = 17 / 14 = 1.21$$

$$\text{MIPS Rating}_2 = 1000 \text{ MHz} / 1.21 = 826.4 \text{ MIPS}$$

$$\text{CPU Time}_2 = ((12 + 1 + 1) \times 10^6 \times 1.21) / (1 \times 10^9) = 0.017 \text{ sec}$$

MIPS rating indicate that compiler 2 is faster

Amadahl's Law

64

- Amadahl's law was established by Gene Amadahl in 1967
- It provides an understanding on scaling, limitations and economics of parallel computing
- Forms a basis for quantitative principles in computer system design.
- It can be used to find the **maximum expected improvement of an overall system when only a part of the system is improved.**
- It basically states the performance improvement to be gained from using some faster mode of execution is limited by a fraction of the time the faster mode can be used.
- Used by computer designers to enhance only those architectural features that result in reasonable performance improvement.
- Referred to as **quantitative principles in design**

- Amadahl's law demonstrates the *law of diminishing returns*.
- An example:
 - Suppose we are improving a part of the computer system that affects only 25% of the overall task
 - The improvement can be *very little or extremely large*.
 - With "*infinite*" speedup, the 25% of the task can be done in "*zero*" time.
 - Maximum possible speedup = $XT_{\text{orig}} / XT_{\text{new}} = 1 / (1 - 0.25) = 1.33$

- Amadahl's law demonstrates the *law of diminishing returns*.
- An example:
 - Suppose we are improving a part of the computer system that affects only 25% of the overall task
 - The improvement can be *very little or extremely large*.
 - With "*infinite*" speedup, the 25% of the task can be done in "*zero*" time.
 - Maximum possible speedup = $XT_{\text{orig}} / XT_{\text{new}} = 1 / (1 - 0.25) = 1.33$
 - Amadahl's law concerns the speedup achievable from an improvement in computation that affects a fraction **F** of the computation, where the improvement has a Speedup of **S**

Before improvement



After improvement



Before improvement



After improvement



- Execution time before improvement : $(1 - F) + F = 1$
- Execution time after improvement : $(1 - F) + F / S$
- Speedup obtained:

$$Speedup = \frac{1}{(1 - F) + \frac{F}{S}}$$

- As $S \rightarrow \infty$, $Speedup \rightarrow 1 / (1 - F)$

The **fraction F** limits the maximum speedup that can be obtained

Illustration of law of diminishing returns

- Let $F = 0.25$
- The table shows the speedup $= (1 / (1 - F + F / S))$ for various values of S

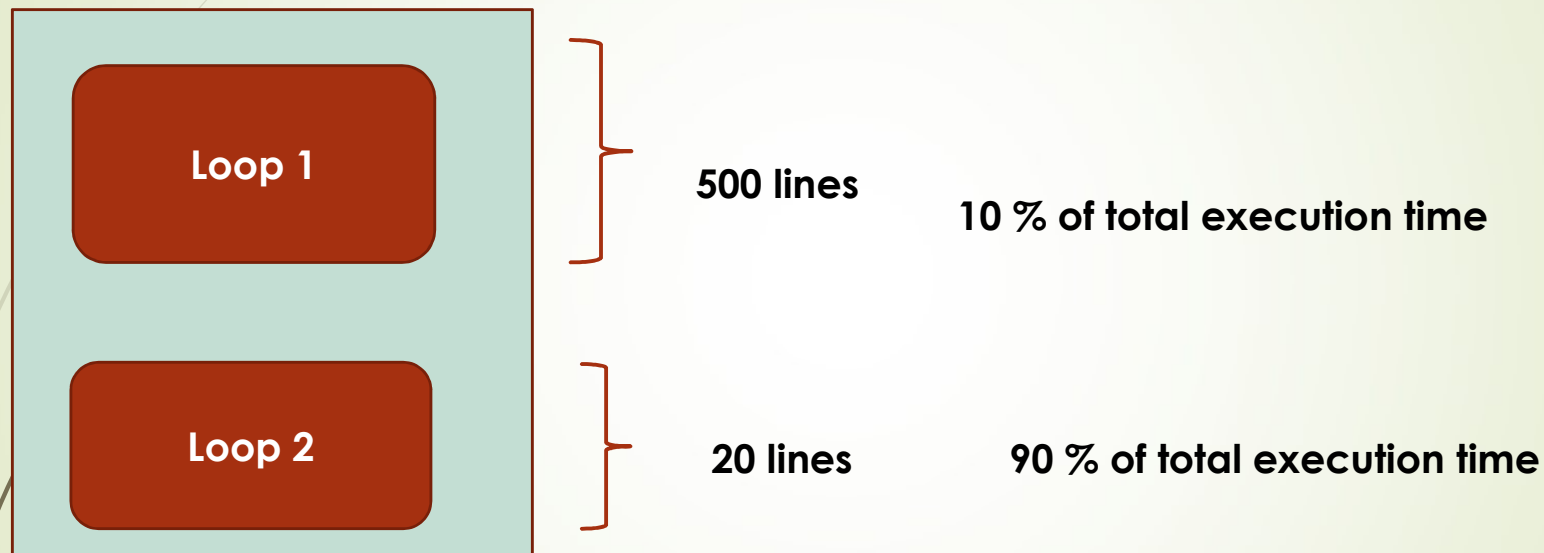
S	Speedup
1	1.00
2	1.14
5	1.25
10	1.29
50	1.32
100	1.33
1000	1.33
100,000	1.33

Illustration of law of diminishing returns

- Let $F = 0.75$
- The table shows the speedup for various values of $S = (1 / (1 - 0.75)) = 4.00$

S	Speedup
1	1.00
2	1.60
5	2.50
10	3.08
50	3.77
100	3.88
1000	3.99
100,000	4.00

Design alternative using Amadahl's law



- We make 10% of a program 90X faster, speedup= $1 / (0.9 + 0.1 / 90) = 1.11$
- We make 90% of a program 10X faster, speedup = $1 / (0.1 + 0.9 / 20) = 5.26$

Example 2

71

- The execution time of a programs on a machine is found to be 50 seconds, out of which 42 seconds is consumed by a multiply operations. It is required to make the program run 5 times faster. By how much must the speed of the multiplier be improved?

Example 2

72

- The execution time of a programs on a machine is found to be 50 seconds, out of which 42 seconds is consumed by a multiply operations. It is required to make the program run 5 times faster. By how much must the speed of the multiplier be improved?

- Here $F = 42 / 50 = 0.84$

- According to Amadahl's law

$$S = 1 / (0.16 + 0.84 / S)$$

$$0.80 + 4.2 / S = 1$$

$$S = 21$$

Example 3

73

- The execution time of a programs on a machine is found to be 50 seconds, out of which 42 seconds is consumed by a multiply operations. It is required to make the program run **8** times faster. By how much must the speed of the multiplier be improved?

Example 3

74

- The execution time of a programs on a machine is found to be 50 seconds, out of which 42 seconds is consumed by a multiply operations. It is required to make the program run **8** times faster. By how much must the speed of the multiplier be improved?

- Here $F = 42 / 50 = 0.84$

- According to Amadahl's law

$$8 = 1 / (0.16 + 0.84 / S)$$

$$1.28 + 6.72 / S = 1$$

$$S = -24$$

No amount to speed improvement in the multiplier can achieve this.

Maximum speedup achievable:

$$1 / (1 - F) = 6.25$$

Example 4

75

- Suppose we plan to upgrade the processor of webserver. The CPU is 30 times faster on search queries than the old processor. The old processor is busy with search queries 80% of the time. Estimate the speedup obtained by the upgrade

Example 4

76

- Suppose we plan to upgrade the processor of webserver. The CPU is 30 times faster on search queries than the old processor. The old processor is busy with search queries 80% of the time. Estimate the speedup obtained by the upgrade
- Here, $F = 0.80$ and $S = 30$
- Thus, $\text{speedup} = 1 / (0.2 + 0.8 / 30) = 4.41$