Computer System Architecture

Prof. Sheetal S Shah

Department of Computer Engineering

Dharmsinh Desai University

- Computer Architecture and Organization by John P. Hayes, Third Edition.
- Computer Organization & Architecture by William Stallings,
- Computer System Architecture by Morris Mano
- Computer organization by Carl Hamacher and Zaky

Architecture & Organization

- Architecture is those attributes visible to the programmer–Instruction set, number of bits used for data representation, I/O mechanisms, addressing techniques.

  –e.g. Is there a multiply instruction?
- Organization is how features are implemented

  –Control signals, interfaces, memory technology.
  –e.g. Is there a hardware multiply unit or is it done by repeated addition?

Layered view of computer design

TOP LAYER	COMPUTER ARCHITECTURE
MIDDLE LAYER	COMPUTER ORGANIZATION
BOTTOM LAYER	COMPUTER LOGIC

In this course the word *computer architecture* is intended to cover the aspects of computer design

- instruction set architecture
- Organization
- hardware

Computing and Computer

- The Nature of Computing
  - The Elements of Computers
  - The brain versus the computer
  - An abstract computer
  - Limitations of Computers
- The Evolution of Computers
  - The Mechanical Computer
  - Electronic Computer
  - First Generation
  - Second Generation
  - Third Generation
  - Fourth Generation (VLSI)

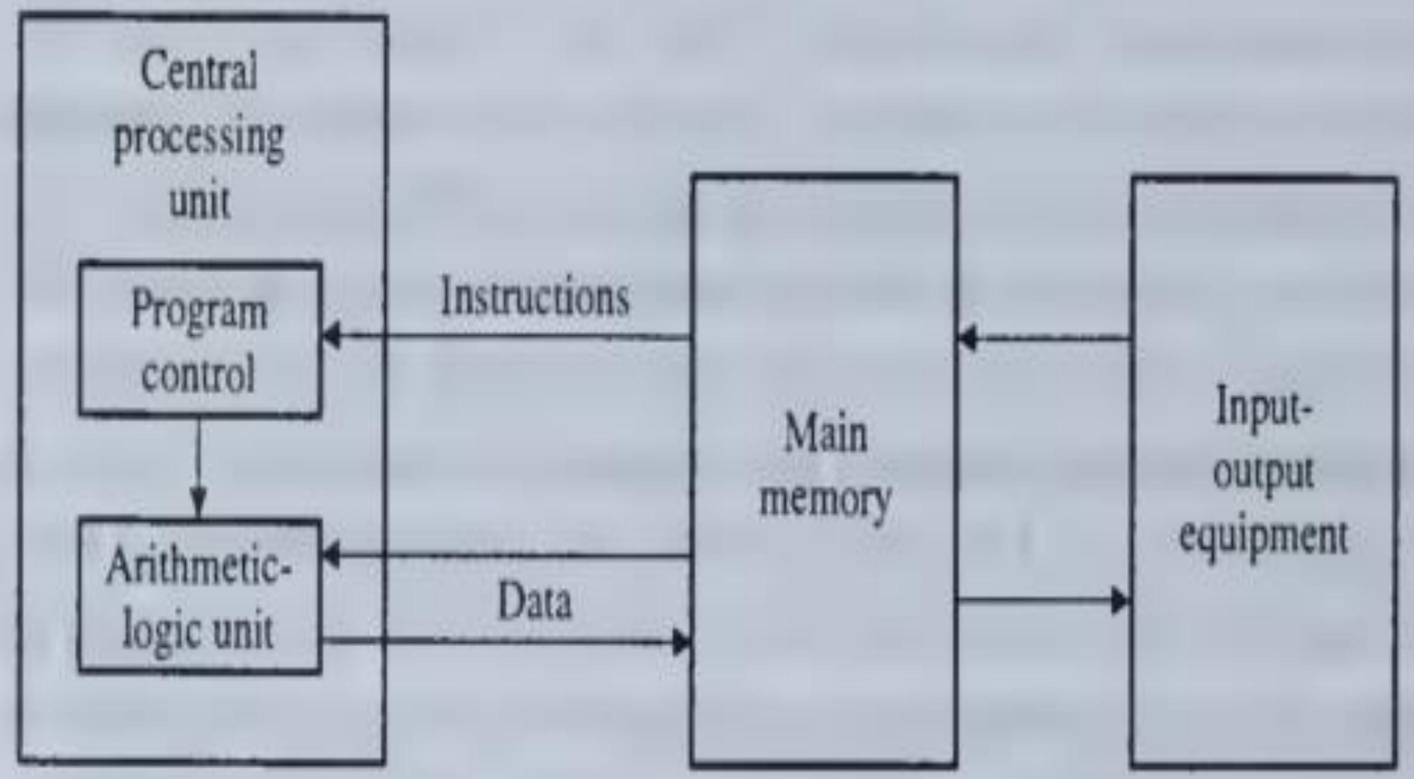
The Nature of Computing

- Throughout history humans have relied mainly on their brains to perform calculation; in other words, they were the computers. As civilization advanced, a variety of computing tools were invented that aided, but did not replace, manual computation.
- The earliest peoples used their fingers, pebbles (stone),or tally sticks for counting purposes.
- The Latin words digitus meaning “finger” and calculus meaning “pebble” have given us digital and calculate and indicate the ancient origins of these computing concepts.

The Nature of Computing

- The early computational aids that were widely used until quite recently are:
  - -The abacus
  - -And slide rule.
- Definition of Abacus:

  An abacus is a mechanical device used to aid an individual in performing mathematical calculations.


Central processing unit
Program control
Arithmetic-logic unit
Instructions
Data
Main memory
Input-output equipment

Abstract Computer

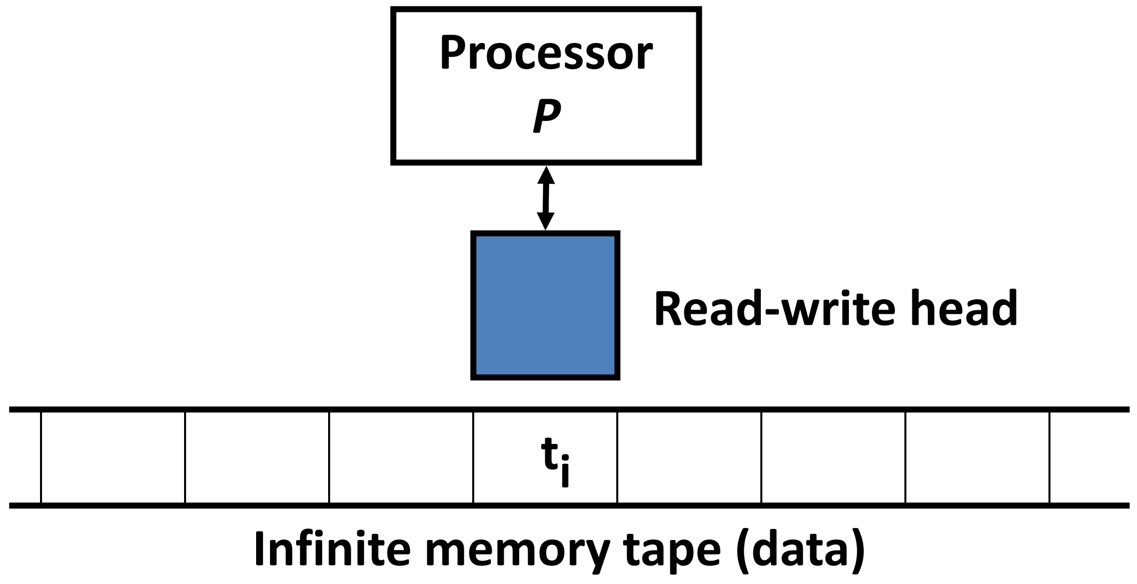
- Are there any computations that a “reasonable” computer can never perform?
- Three notion of reasonableness are
  - Computer should not store answers to all possible problems.
  - Computer should only be required to solve the problem for which a solution procedure or program can be given.
  - Computer should process information at a finite speed.

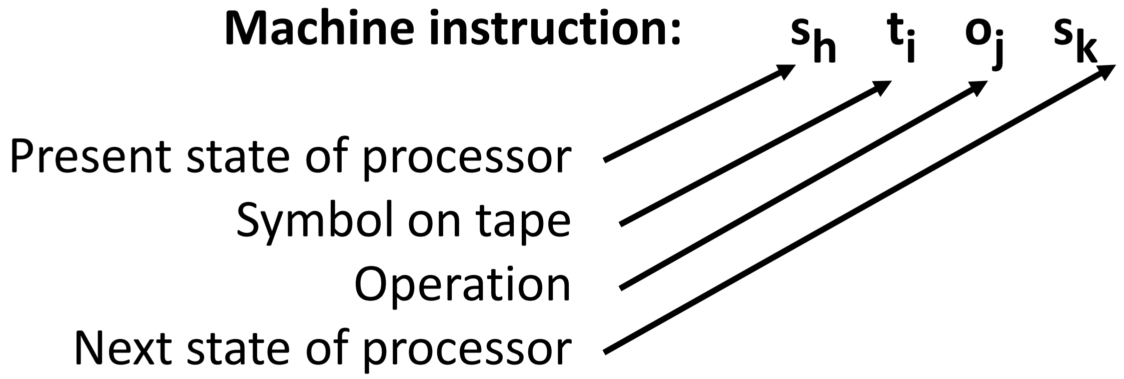
Theory of Computing

- Alan Turing (1912-1954) gave a model of computing in 1936 – *Turing Machine*.
- Original paper: A. M. Turing, "On Computable Numbers with an Application to the *Entscheidungsproblem*," Proc. Royal Math. Soc.*, ser. 2, vol. 42, pp. 230-265, 1936.
- Recent book: David Leavitt, *The Man Who Knew Too Much: Alan Turing and the Invention of the Computer (Great Discoveries),* W. W. Norton & Co., 2005.

* *The question of decidability, posed by mathematician Hilbert.*

Turing Machine




Machine instruction: s_h t_i o_j s_k
Present state of processor
Symbol on tape
Operation
Next state of processor

Turing Machine

- Four operations:
  - $o_j = t_j$, replace present symbol t_i by t_j
  - $o_j = R$, move head one position to right
  - $o_j = L$, move head one position to left
  - $o_j = H$, halt the computation
- Real computers have finite memory – they find certain problems *intractable*.

Ref: J. P. Hayes, *Computer Architecture and Organization*, New York: McGraw-Hill, 1978.

Example

- Start with a blank tape and create a pattern 0b1b0b1b . . .
- Define symbols: b (blank), 0, 1

Present state	Symbol on tape	Operation	Next state
S0 (begin)	blank	Write 0 and move right	S1
S1	blank	Move right	S2
S2	blank	Write 1 and move right	S3
S3	blank	Move right	S0

First General-Purpose Computer

- Electronic Numerical Integrator and Calculator (ENIAC) built in World War II was the first general purpose computer
  - Used for computing artillery firing tables
  - 80 feet long, 8.5 feet high and several feet wide
  - Twenty 10 digit registers, each 2 feet long
  - Used 18,000 vacuum tubes
  - 5,000 additions/second
  - Weight: 30 tons
  - Power consumption: 140kW

Univac: Election, Nov. 4, 1952

Candidate	Electoral votes	
	Univac prediction	Actual count
Eisenhower	438	442
Stevenson	93	89

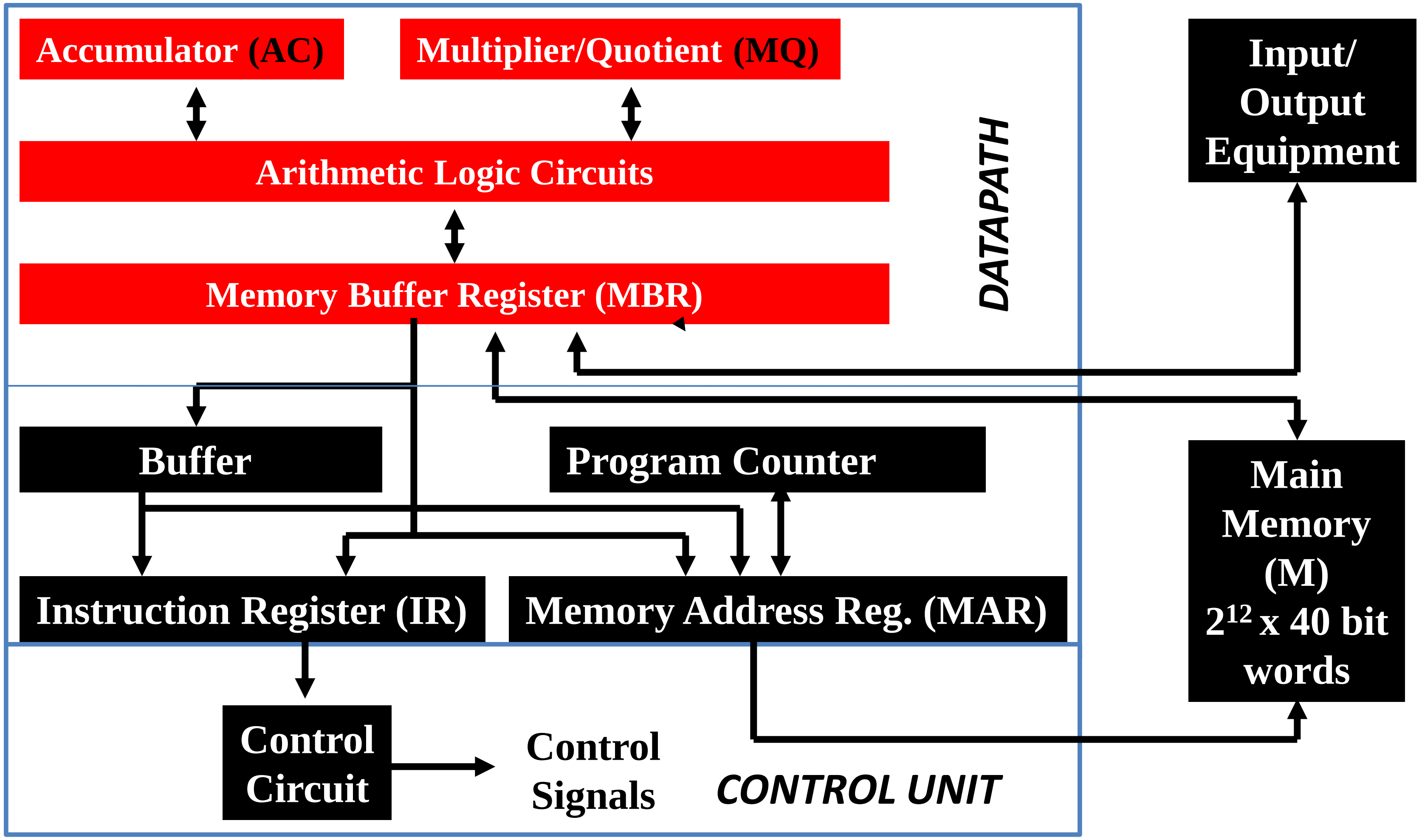
Harold Sweeney, operator
J. Presper Eckert, co-inventor
Walter Cronkite, CBS

USA TODAY, Oct 27, 2004, p. B.3

First-Generation Computers

- Late 1940s and 1950s
- Stored-program computers
- Programmed in assembly language
- Used magnetic devices and earlier forms of memories
- Examples: IAS, ENIAC, EDVAC, UNIVAC, Mark I, IBM 701
- Each instruction of EDVAC had the form
  - A1 A2 A3 A4 OP

The Organization of IAS Computer

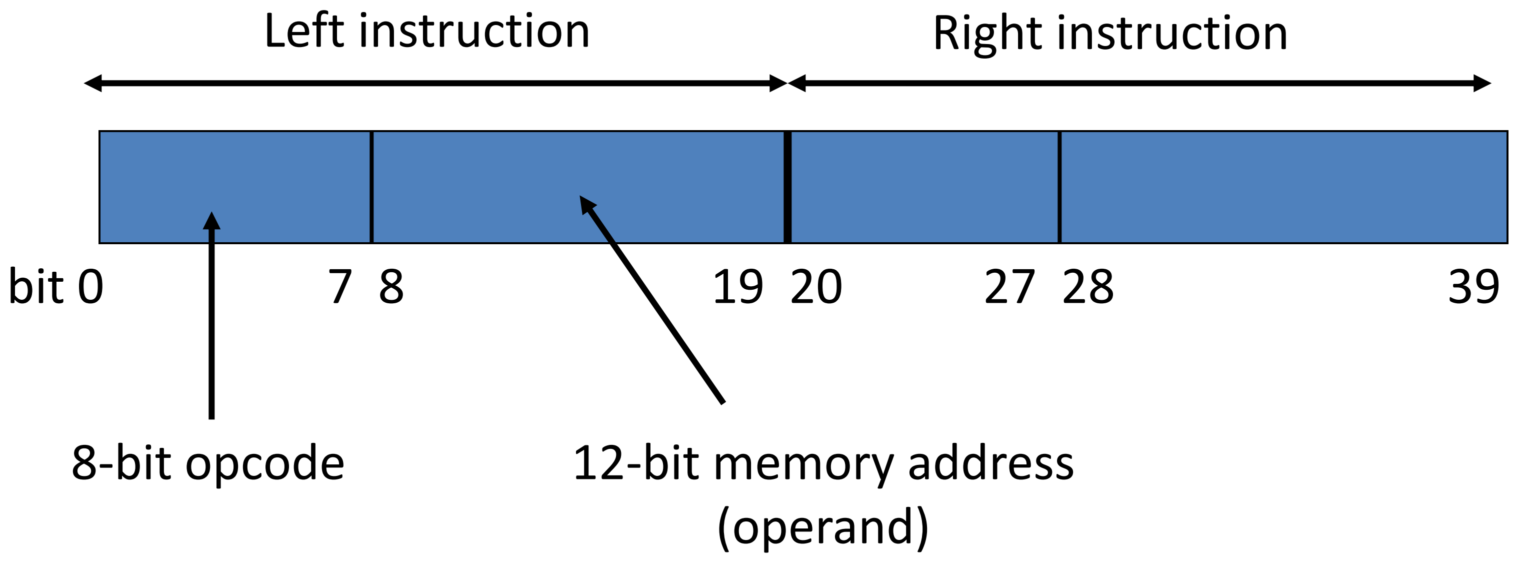


Accumulator (AC)
Multiplier/Quotient (MQ)
Arithmetic Logic Circuits
Memory Buffer Register (MBR)
DATAPATH
Input/ Output Equipment
Buffer
Program Counter
Instruction Register (IR)
Memory Address Reg. (MAR)
Main Memory (M) 2^{12} x 40 bit words
Control Circuit
Control Signals
CONTROL UNIT

17

IAS Computer Machine Language

40-bit word, two machine instructions per word.



Ref: **J. P. Hayes, *Computer Architecture and Organization*, New York: McGraw-Hill, 1978.**

IAS Data Transfer Instructions (7)

Instruction	*Opcode*	*Description*
• LOAD MQ	00001010	AC ← MQ
• LOAD MQ, M(X)	00001001	MQ ← M(X)
• STOR M(X)	00100001	M(X) ← AC
• LOAD M(X)	00000001	AC ← M(X)
• LOAD – M(X)	00000010	AC ← – M(X)
• LOAD |M(X)|	00000011	AC ← |M(X)|
• LOAD – |M(X)|	00000100	AC ← – |M(X)|

IAS Unconditional Branch Instructions (2)

Instruction	*Opcode*	*Description*
• JUMP M(X,0:19)	00001101	next instruction M(X,0:19)
• JUMP M(X,20:39)	00001110	next instruction M(X,20:39)

IAS Conditional Branch Instructions (2)

Instruction	*Opcode*	*Description*
JUMP +M(X,0:19)	00001111	IF AC ≥ 0, then next instruction M(X,0:19)
JUMP +M(X,20:39)	00010000	IF AC ≥ 0, then next instruction M(X,20:39)

IAS Arithmetic Instructions (8)

Instruction	*Opcode*	*Description*
• ADD M(X)	00000101	AC ← AC + M(X)
• ADD |M(X)|	00000111	AC ← AC + |M(X)|
• SUB M(X)	00000110	AC ← AC – M(X)
• SUB |M(X)|	00001000	AC ← AC – |M(X)|
• MUL M(X)	00001011	AC, MQ ← MQ×M(X)
• DIV M(X)	00001100	MQ, AC ← MQ/M(X)
• LSH	00010100	AC ← AC x 2
• RSH	00010101	AC ← AC / 2

IAS Address Modify Instructions (2)

Instruction	*Opcode*	*Description*
STOR M(X,8:19)	00010010	M(X,8:19) ← AC(28:39)
STOR M(X,28:39)	00010011	M(X,28:39) ← AC(28:39)

How IAS Computer Adds Two Numbers

- Suppose the numbers are stored in memory locations 100 and 101, and
- The sum is to be saved in memory location 102

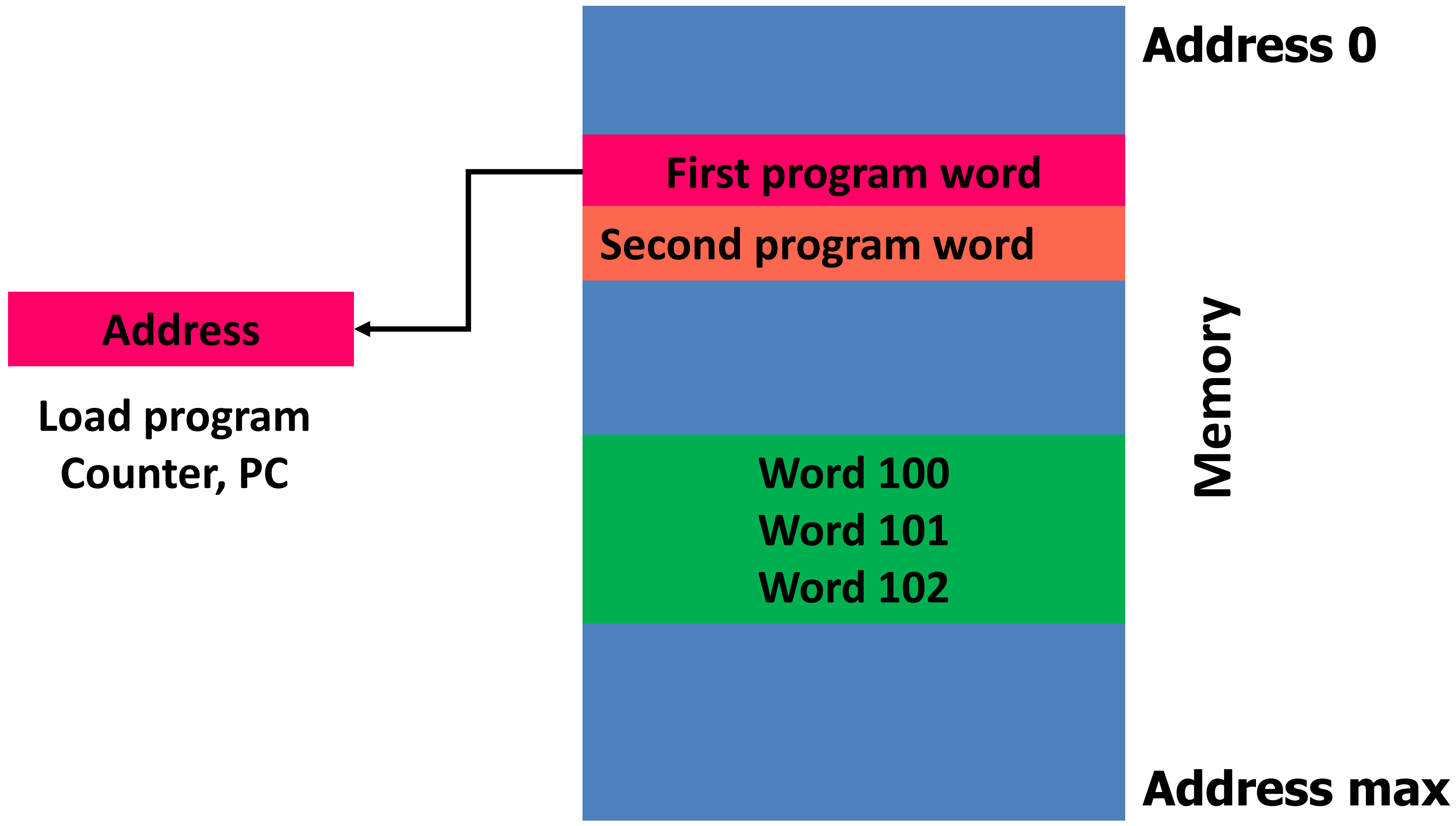
Instruction	***Opcode***	***Description***
LOAD M(100)	00000001	AC ← M(100)
ADD M(101)	00000101	AC ← AC + M(101)
STOR M(102)	00100001	M(102) ← AC

IAS Computer Machine Code

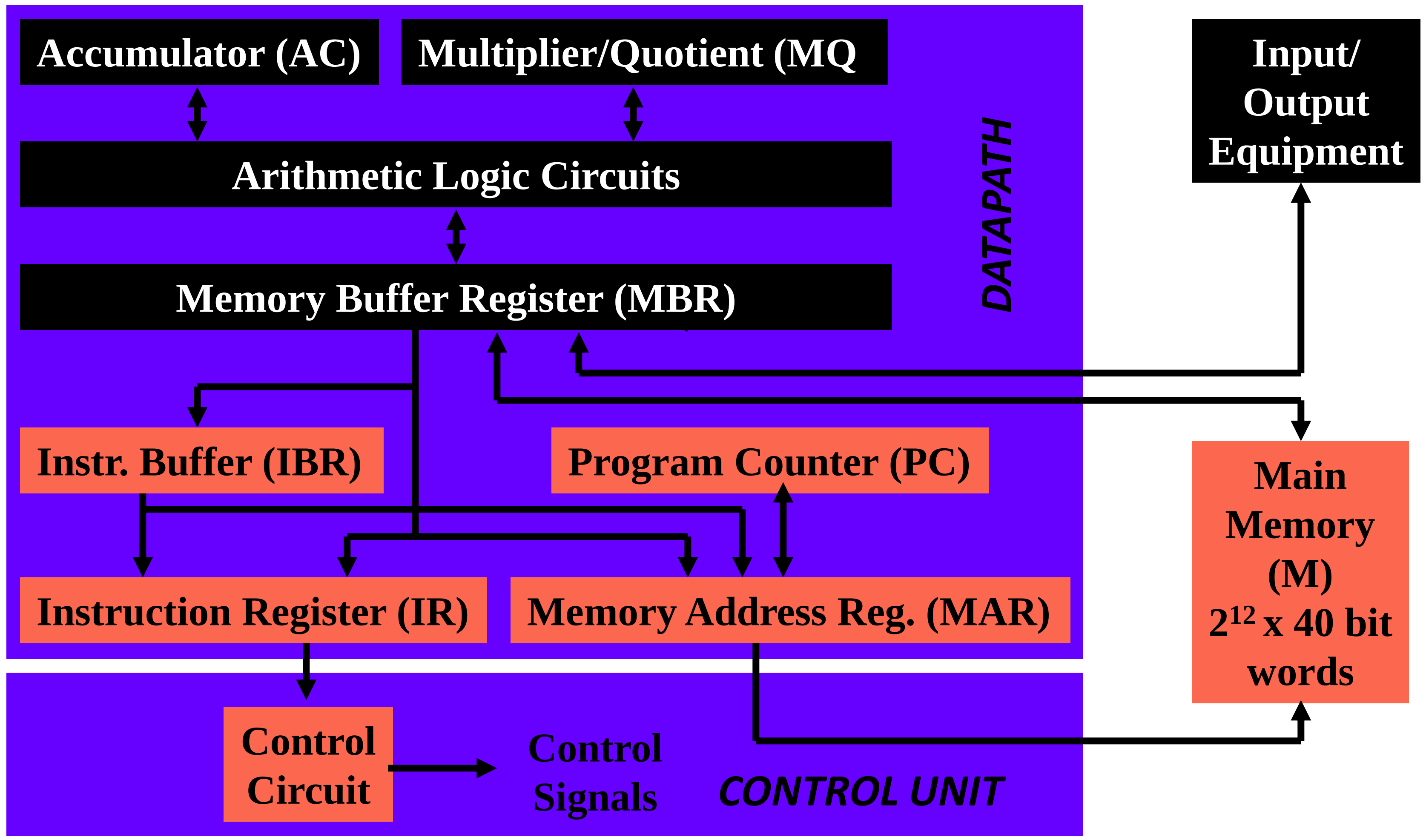
00000001	000001100100	00000101	000001100101
Load	100	Add	101

00100001	000001100110	00000000	000000000000
Stor	102	Stop	

Save Program in Memory



Executing the Program



IAS Instruction Cycles

- Store machine code in contiguous words of memory.
- Place starting address in program counter (PC).
- Start program: MAR ← PC
- Read memory: IBR ← MBR ← M(MAR), *fetch*
- Place left instruction (Load) in IR and operand (address) 100 in MAR, *decode*
- Read memory: AC ← M(100), *execute*
- Place right instruction (Add) in IR and operand (address) 101 in MAR, *decode*
- Read memory and add: AC ← AC + M(101), *execute*
- PC ← PC + 1

IAS Instruction Cycles (Cont.)

- MAR ← PC
- Read memory: IBR ← MBR ← M(MAR), *fetch*
- Place left instruction (Stor) in IR and operand (address) 102 in MAR, *decode*
- MBR ← AC, *execute*
- Write memory
- Place right instruction (Stop) in IR and operand 000 in MAR, *decode*
- Stop, *execute*

Hardware Contains

- Data storage devices
    - Memory
    - Registers
- Instruction decoding and execution devices
    - Execution unit (arithmetic logic unit, ALU)
    - Data transfer buses
    - Control unit

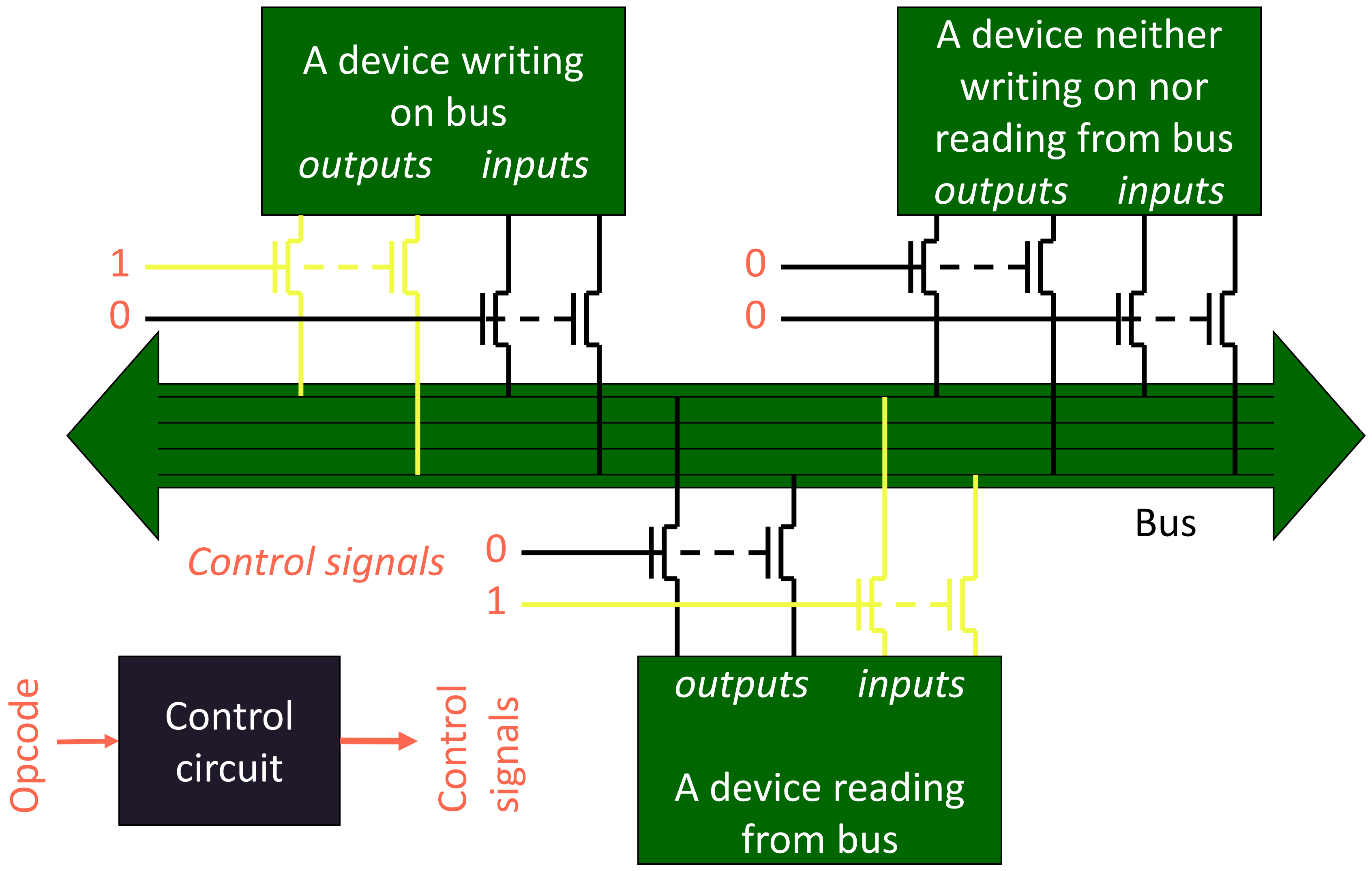
Registers in IAS

Register	Size (bits)	Function
Program counter (PC)	12	Holds mem. address of next instruction
Accumulator (AC)	40	Temporary data storage
Multiplier quotient (MQ)	40	Temporary data storage
Memory buffer (MBR)	40	Memory read / write data
Instruction buffer (IBR)	20	Holds right instr. (bits 20-39)
Instruction register (IR)	8	Holds opcode part of instruction
Memory address (MAR)	12	Holds mem. address part of instruction

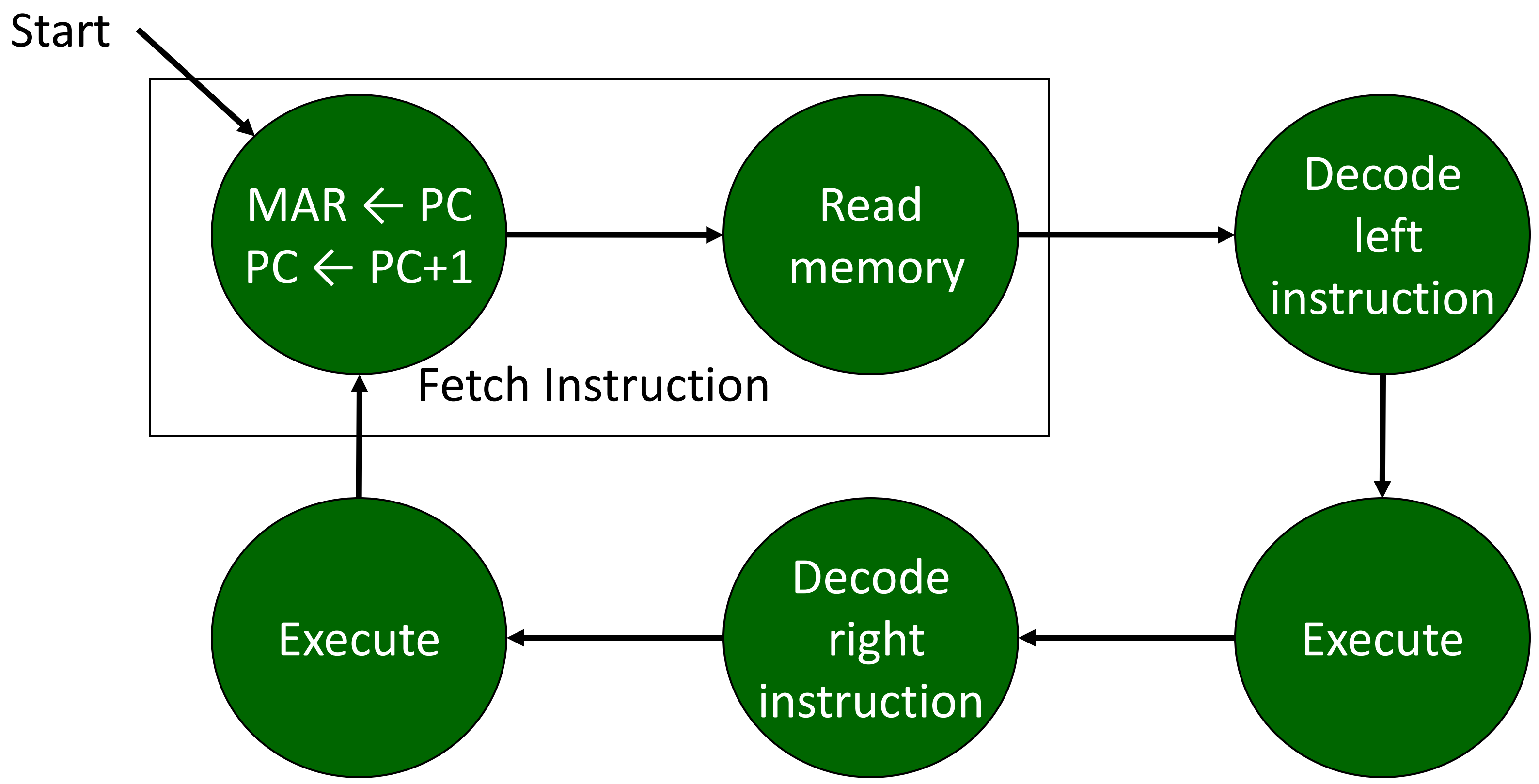
Register Transfer

- Transfer data synchronously with clock
    - Register to register
    - Register to register through ALU logic
    - Registers to register through memory (write)
    - Register to register through memory (read)
- Data transfer through communication bus
    - Source register writes on bus
    - Destination register reads from bus
    - Control circuit provides read / write signals for bus and memory

Communication Bus



Control Circuit – Finite State Machine



Von Neumann Bottleneck

- Von Neumann architecture uses the same memory for instructions (program) and data.
- The time spent in memory accesses can limit the performance. This phenomenon is referred to as *von Neumann bottleneck.*
- To avoid the bottleneck, later architectures restrict most operands to registers (temporary storage in processor).

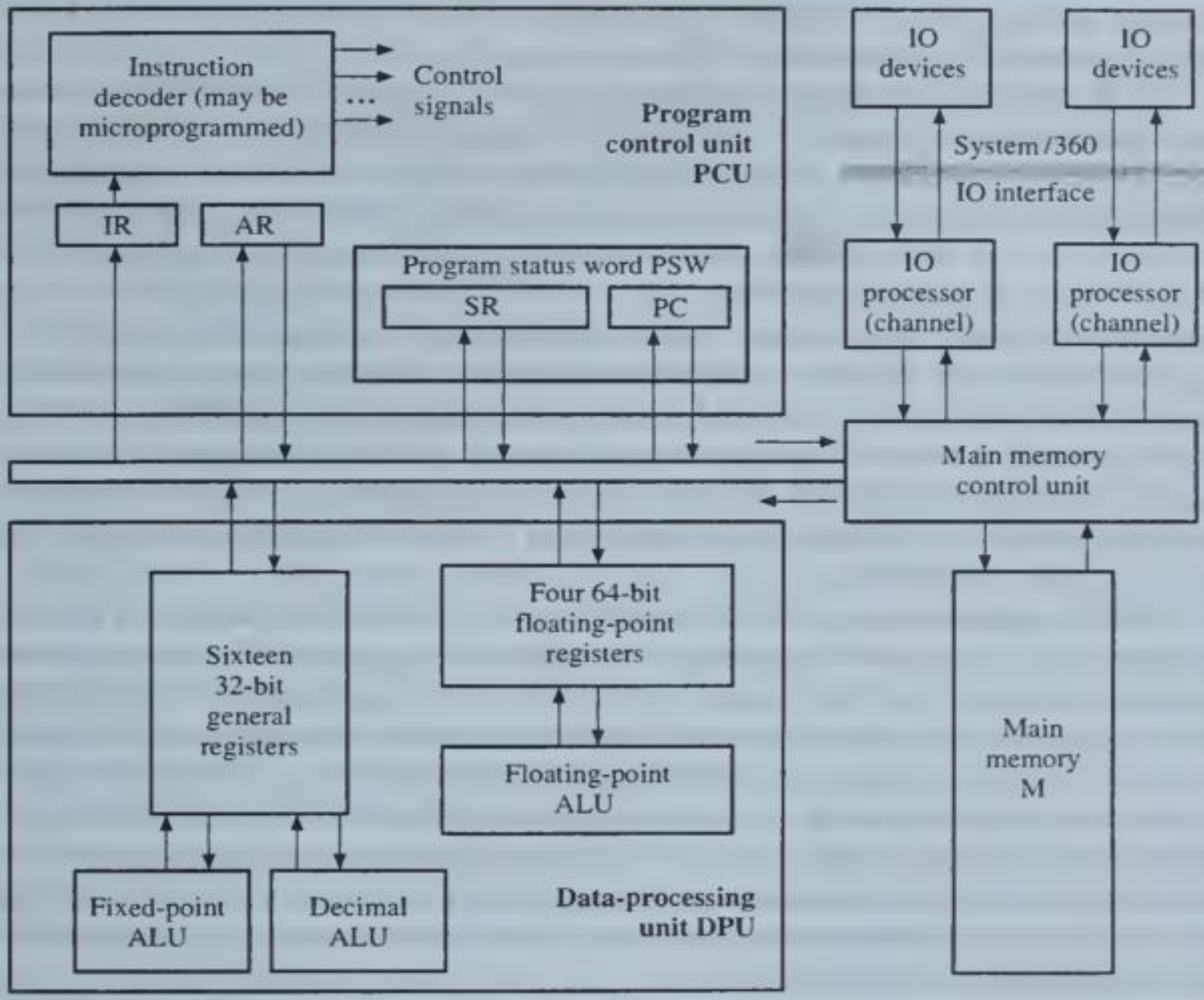
Ref.: D. E. Comer, *Essentials of Computer Architecture*, Upper Saddle River, NJ: Pearson Prentice-Hall, 2005, p. 87.

Second Generation Computers

- 1955 to 1964
- Transistor replaced vacuum tubes
- Magnetic core memories
- Floating-point arithmetic
- High-level languages used: ALGOL, COBOL and FORTRAN
- System software: compilers, subroutine libraries, batch processing
- Example: IBM 7094

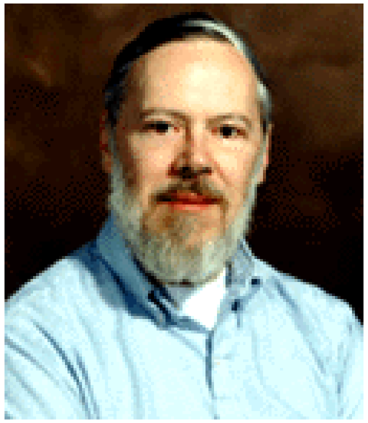
Third Generation Computers

- Beyond 1965
- Integrated circuit (IC) technology
- Semiconductor memories
- Memory hierarchy, virtual memories and caches
- Time-sharing
- Parallel processing and pipelining
- Microprogramming
- Examples: IBM 360 and 370, CYBER, ILLIAC IV, DEC PDP and VAX, Amdahl 470


Instruction
decoder (may be
microprogrammed)
Control
signals
Program
control unit
PCU
IR
AR
Program status word PSW
SR
PC
IO
devices
IO
devices
System/360
IO interface
IO
processor
(channel)
IO
processor
(channel)
Main memory
control unit
Sixteen
32-bit
general
registers
Four 64-bit
floating-point
registers
Floating-point
ALU
Fixed-point
ALU
Decimal
ALU
Data-processing
unit DPU
Main
memory
M

C Programming Language and UNIX Operating System

1972



Now

Speedup techniques

Number of speed enhancing features have been incorporated into the design of computers

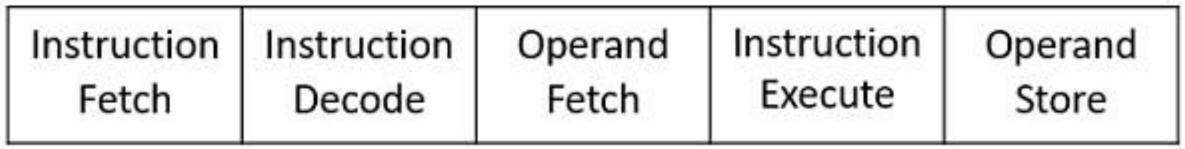
1.Introducing the Cache memory

2.Pipelining

3.Superscalar Processing

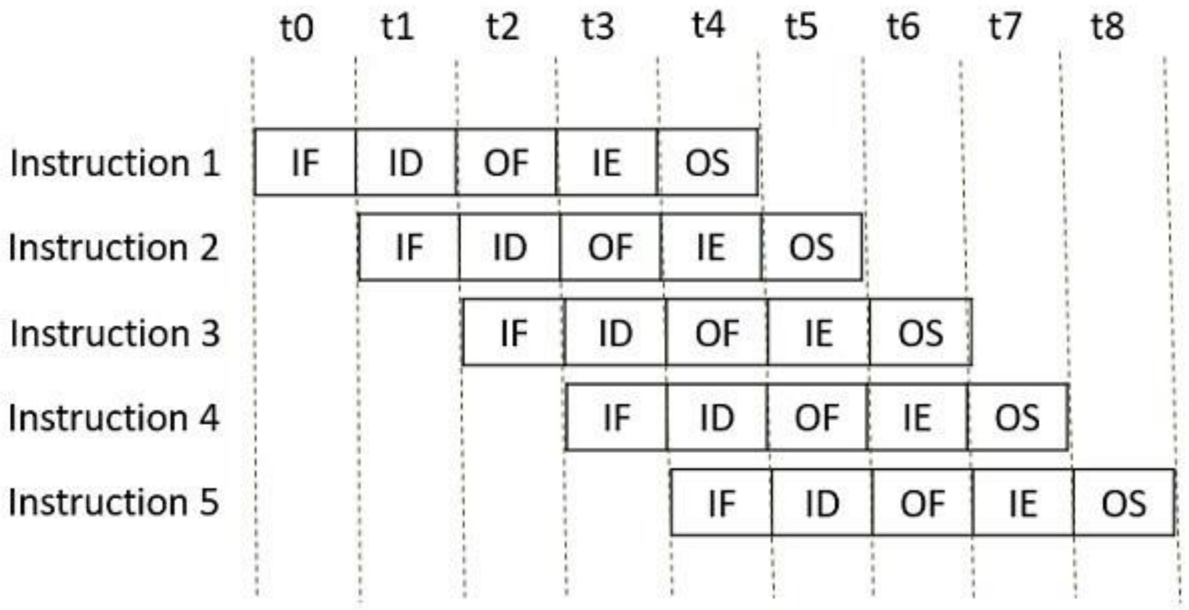
Pipelining

- An instruction in a process is divided into 5 subtasks likely,



Instruction Fetch	Instruction Decode	Operand Fetch	Instruction Execute	Operand Store

•In the first subtask, the instruction is fetched.
•The fetched instruction is decoded in the second stage.
•In the third stage, the operands of the instruction are fetched.
•In the fourth, arithmetic and logical operation are performed on the operands to execute the instruction.
•In the fifth stage, the result is stored in memory.



Pipelining of 5 Instructions

The New Generation

- Personal computers
- Laptops and Palmtops
- Networking and wireless
- SOC and MEMS technology
- And the future!
    - Biological computing
    - Molecular computing
    - Nanotechnology
    - Optical computing
    - Quantum computing
    - See articles listed on the next slide and available at E7700: Advanced VLSI Design course site,

Performance Measure

- A user of a computer measures its performance based on the time taken to execute a given job (program).
- On the other hand, a laboratory engineer measures the performance of his system by the total amount of work done in a given time.
- To maximize performance, we want to minimize response time or execution time for some task. Thus, we can relate performance and execution time for a computer X:

$$\text{Performance}_X = 1/\text{Execution time}_X$$

- Time can be defined in different ways, depending on what we are measuring:
  - Response time : The time between the start and completion of a task. It includes time spent executing on the CPU, accessing disk and memory, waiting for I/O and other processes, and operating system overhead. This is also referred to as **execution time.**
  - Throughput :The total amount of work done in a given time.
  - CPU execution time : Total time a CPU spends computing on a given task (excludes time for I/O or running other programs). This is also referred to as simply CPU time.

- Let the number of CPU clock cycles for executing a job to be the cycle count (CC), the cycle time by CT and the clock frequency by f = 1/CT.
- The time taken by the CPU to execute a job can be expressed as
  – CPU Time=CC*CT= CC/f

- It may be easier to count the number of instructions executed in a given program as compared to counting the number of CPU clock cycles needed for executing that program.
- Therefore, the average number of clock cycles per instruction (CPI) has been used as an alternate performance measure. The following equation shows how to compute the CPI.

$$CPI = \frac{CPU\ clock\ cycles\ for\ the\ program}{Instruction\ count}$$

$$CPU\ time = Instruction\ count \times CPI \times Clock\ cycle\ time$$

$$= \frac{Instruction\ count \times CPI}{Clock\ rate}$$

The instruction set of a given machine consists of a number of instruction categories: ALU (simple assignment and arithmetic and logic instructions), load, store, branch, and so on. In the case that the CPI for each instruction category is known, the overall CPI can be computed as

$$CPI = \frac{\sum_{i=1}^{n} CPI_i \times I_i}{Instruction\ count}$$

I_i is the number of times an instruction of type i is executed in the program.
CPI_i is the average number of clock cycles needed to execute such instruction.

Example

- Let assume that a benchmark has 100 instructions:
  - 25 instructions are loads/stores (each take 2 cycles)
  - 50 instructions are adds (each takes 1 cycle)
  - 25 instructions are square root (each takes 50 cycles)

  What is the CPI for this benchmark?

CPI = ((0.25 * 2) + (0.50 * 1) + (0.25 * 50)) = 13.5

Example 1

- A program runs in 10 seconds on computer A, which has a 2 GHz clock. We are trying to help a computer designer build a computer, B, which will run this program in 6 seconds. The designer has determined that a substantial increase in the clock rate is possible, but this increase will affect the rest of the CPU design, causing computer B to require 1.2 times as many clock cycles as computer A for this program. What clock rate should we tell the designer to target?

Let's first find the number of clock cycles required for the program on A:

$$\text{CPU time}_{\text{A}} = \frac{\text{CPU clock cycles}_{\text{A}}}{\text{Clock rate}_{\text{A}}}$$

$$10 \text{ seconds} = \frac{\text{CPU clock cycles}_{\text{A}}}{2 \times 10^9 \ \dfrac{\text{cycles}}{\text{second}}}$$

$$\text{CPU clock cycles}_{\text{A}} = 10 \text{ seconds} \times 2 \times 10^9 \ \frac{\text{cycles}}{\text{second}} = 20 \times 10^9 \text{ cycles}$$

CPU time for B can be found using this equation:

$$\text{CPU time}_{\text{B}} = \frac{1.2 \times \text{CPU clock cycles}_{\text{A}}}{\text{Clock rate}_{\text{B}}}$$

$$6 \text{ seconds} = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{\text{Clock rate}_{\text{B}}}$$

$$\text{Clock rate}_{\text{B}} = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{6 \text{ seconds}} = \frac{0.2 \times 20 \times 10^9 \text{ cycles}}{\text{second}} = \frac{4 \times 10^9 \text{ cycles}}{\text{second}} = 4 \text{ GHz}$$

To run the program in 6 seconds, B must have twice the clock rate of A.

Example 2

- Suppose we have two implementations of the same instruction set architecture. Computer A has a clock cycle time of 250 ps and a CPI of 2.0 for some program, and computer B has a clock cycle time of 500 ps and a CPI of 1.2 for the same program. Which computer is faster for this program and by how much?

- First, find the number of processor clock cycles for each computer:

$$\begin{aligned}\text{CPU clock cycles}_A &= I \times 2.0\\ \text{CPU clock cycles}_B &= I \times 1.2\end{aligned}$$

Now we can compute the CPU time for each computer:

$$\begin{aligned}\text{CPU time}_A &= \text{CPU clock cycles}_A \times \text{Clock cycle time}\\ &= I \times 2.0 \times 250\text{ ps} = 500 \times I\text{ ps}\end{aligned}$$

Likewise, for B:

$$\text{CPU time}_B = I \times 1.2 \times 500\text{ ps} = 600 \times I\text{ ps}$$

Clearly, computer A is faster. The amount faster is given by the ratio of the execution times:

$$\frac{\text{CPU performance}_A}{\text{CPU performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = \frac{600 \times I\text{ ps}}{500 \times I\text{ ps}} = 1.2$$

Example 3

- A compiler designer is trying to decide between two code sequences for a particular computer. The hardware designers have supplied the following facts:

	CPI for each instruction class		
	A	B	C
CPI	1	2	3

For a particular high-level language statement, the compiler writer is considering two code sequences that require the following instruction counts:

	Instruction counts for each instruction class		
Code sequence	A	B	C
1	2	1	2
2	4	1	1

Which code sequence executes the most instructions? Which will be faster? What is the CPI for each sequence?

i. Sequence 1 executes 5 instructions and sequence 2 executes 6 instructions

$$\text{CPU clock cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times C_i)$$

This yields

$$\text{CPU clock cycles}_1 = (2 \times 1) + (1 \times 2) + (2 \times 3) = 2 + 2 + 6 = 10 \text{ cycles}$$

$$\text{CPU clock cycles}_2 = (4 \times 1) + (1 \times 2) + (1 \times 3) = 4 + 2 + 3 = 9 \text{ cycles}$$

$$\text{CPI} = \frac{\text{CPU clock cycles}}{\text{Instruction count}}$$

$$\text{CPI}_1 = \frac{\text{CPU clock cycles}_1}{\text{Instruction count}_1} = \frac{10}{5} = 2.0$$

$$\text{CPI}_2 = \frac{\text{CPU clock cycles}_2}{\text{Instruction count}_2} = \frac{9}{6} = 1.5$$

- Always it should be noted that the only complete and reliable measure of computer performance is time.
  - For example, changing the instruction set to lower the instruction count may lead to an organization with a slower clock cycle time or higher CPI that offsets the improvement in instruction count.
  - Similarly, because CPI depends on type of instructions executed, the code that executes the fewest number of instructions may not be the fastest.

$$\text{Time} = \text{Seconds/Program} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

Components of performance	Units of measure
CPU execution time for a program	Seconds for the program
Instruction count	Instructions executed for the program
Clock cycles per instruction (CPI)	Average number of clock cycles per instruction
Clock cycle time	Seconds per clock cycle

Some other Performance Metric

- MIPS : millions of instructions per second

  MIPS = instruction count / (execution time x 10^6)

- MFLOPS : millions of floating point operations per second

  MFLOPS = floating point operations / (execution time x10^6)

THANK YOU