CPE 343
Computer Organization & Architecture

Moazzam Ali Sahi

Room # B1, B-Block

COMSATS University Islamabad, Lahore Campus

Introduction

- These slides discusses
  - The Computer Hardware
  - Software
  - Interconnection
- It also discusses concepts like
  - Computer Types
  - Evolution of computers
  - Functional units
  - Basic operations
  - RISC and CISC systems.

UNIT-I INTRODUCTION

- Evolution of Computer Systems
- Computer Types
- Functional units
- Basic operational concepts
- Bus structures
- Memory location and addresses
- Memory operations
- Addressing modes
- Design of a computer system
- Instruction and instruction sequencing
- RISC versus CISC

Brief History of Computer Evolution

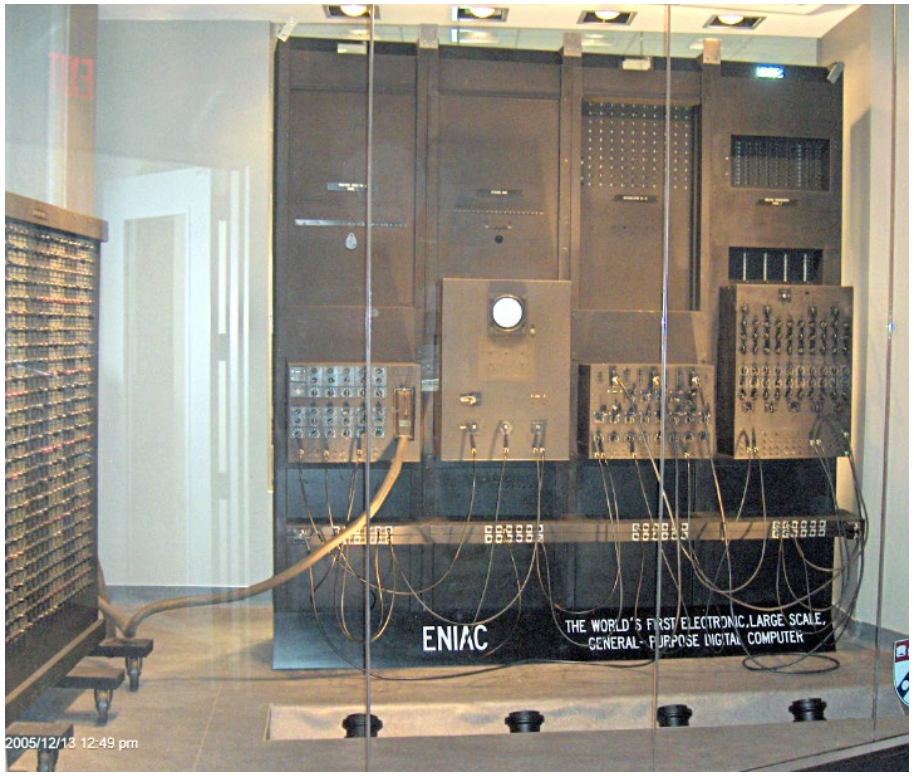
Two phases:

- Before VLSI 1945 –1978
  - ENIAC (Electronic Numerical Integrator And Computer)
  - IAS (Institute for Advanced Study)
  - IBM (International Business Machines)
  - PDP-8 (Programmed Data Processor)

- VLSI 1978 →present day
  - Microprocessors !

Brief History of Computer Evolution

- **ENIAC** (Electronic Numerical Integrator and Computer)



100 K Hz Clock
17500 Vacuum Tubes
30 ton weight

Evolution of Computers

FIRST GENERATION (1945 –1955)

- Program and data reside in the same memory (stored program concepts –John von Neumann)
- ALP was made used to write programs
- Vacuum tubes were used to implement the functions (ALU & CU design)
- Magnetic core and magnetic tape storage devices are used
- Using electronic vacuum tubes, as the switching components

Evolution of Computers

SECOND GENERATION (1955 –1965)

- Transistor were used to design ALU & CU
- HLL is used (FORTRAN)
- To convert HLL to MLL compiler were used
- Separate I/O processor were developed to operate in parallel with CPU, thus improving the performance
- Invention of the transistor which was faster, smaller and required considerably less power to operate

Evolution of Computers

THIRD GENERATION (1965-1975)

- IC technology improved
- Improved IC technology helped in designing low-cost, high-speed processor and memory modules
- Multiprogramming, pipelining concepts were incorporated
- DOS allowed efficient and coordinate operation of computer system with multiple users
- Cache and virtual memory concepts were developed
- More than one circuit on a single silicon chip became available

Evolution of Computers

FOURTH GENERATION (1975-1985)

- CPU –Termed as microprocessor
- INTEL, MOTOROLA, TEXAS, NATIONAL semiconductors started developing microprocessor
- Workstations, microprocessor (PC) & Notebook computers were developed
- Interconnection of different computer for better communication LAN, MAN, WAN
- Computational speed increased by 1000 times
- Specialized processors like Digital Signal Processor were also developed

Evolution of Computers

BEYOND THE FOURTH GENERATION(1985 –TILL DATE)

- E-Commerce, E-banking, home office
- ARM, AMD, INTEL, MOTOROLA
- High speed processor -GHz speed
- Because of submicron IC technology lot of added features in small size

Brief History

- Vacuum Tubes
- 1947-Point Contact Transistor
- 1948-Bipolar Junction Transistor
- 1958-BJT based Integrated Circuit (Prototype)
- pMOSFET
- 1970s-nMOSFET
- 1980s-CMOS Technology
- 2011-FinFET

Processor	Transistor count	Date of introduction	Manufacturer	Process	Area
Intel 4004	2,300	1971	Intel	10 µm	12 mm²
Intel 8008	3,500	1972	Intel	10 µm	14 mm²
MOS Technology 6502	3,510	1975	MOS Technology	8 µm	21 mm²
Motorola 6800	4,100	1974	Motorola	6 µm	16 mm²
Intel 8080	4,500	1974	Intel	6 µm	20 mm²
RCA 1802	5,000	1974	RCA	5 µm	27 mm²
Intel 8085	6,500	1976	Intel	3 µm	20 mm²
Zilog Z80	8,500	1976	Zilog	4 µm	18 mm²
Motorola 6809	9,000	1978	Motorola	5 µm	21 mm²
Intel 8086	29,000	1978	Intel	3 µm	33 mm²
Intel 8088	29,000	1979	Intel	3 µm	33 mm²
Intel 80186	55,000	1982	Intel	3 µm	
Motorola 68000	68,000	1979	Motorola	4 µm	44 mm²
Intel 80286	134,000	1982	Intel	1.5 µm	49 mm²
Intel 80386	275,000	1985	Intel	1.5 µm	104 mm²
ARM 1	25,000	1985	Acorn		
ARM 2	25,000	1986	Acorn		
Intel 80486	1,180,235	1989	Intel	1 µm	173 mm²
ARM 3	300,000	1989	Acorn		
R4000	1,350,000	1991	MIPS	1.0 µm	213 mm²
ARM 6	30,000	1991	ARM		
Pentium	3,100,000	1993	Intel	0.8 µm	294 mm²
ARM 7	578977	1994	ARM		68.51 mm²
Pentium Pro	5,500,000	1995	Intel	0.5 µm	307 mm²
AMD K5	4,300,000	1996	AMD	0.5 µm	251 mm²
Pentium II	7,500,000	1997	Intel	0.35 µm	195 mm²
AMD K6	8,800,000	1997	AMD	0.35 µm	162 mm²
Pentium III	9,500,000	1999	Intel	0.25 µm	128 mm²
AMD K6-III	21,300,000	1999	AMD	0.25 µm	118 mm²
AMD K7	22,000,000	1999	AMD	0.25 µm	184 mm²
Pentium 4	42,000,000	2000	Intel	180 nm	217 mm²
Atom	47,000,000	2008	Intel	45 nm	24 mm²
Barton	54,300,000	2003	AMD	130 nm	101 mm²

Processor	Transistor count	Date of introduction	Manufacturer	Process	Area
AMD K8	105,900,000	2003	AMD	130 nm	193 mm²
Itanium 2 McKinley	220,000,000	2002	Intel	180 nm	421 mm²
Cell	241,000,000	2006	Sony/IBM/Toshiba	90 nm	221 mm²
Core 2 Duo	291,000,000	2006	Intel	65 nm	143 mm²
Itanium 2 Madison 6M	410,000,000	2003	Intel	130 nm	374 mm²
AMD K10 quad-core 2M L3	463,000,000	2007	AMD	65 nm	283 mm²
ARM Cortex-A9	26,000,000	2007	ARM		
AMD K10 quad-core 6M L3	758,000,000	2008	AMD	45 nm	258 mm²
Itanium 2 with 9MB cache	592,000,000	2004	Intel	130 nm	432 mm²
Core i7	731,000,000	2008	Intel	45 nm	263 mm²
POWER6	789,000,000	2007	IBM	65 nm	341 mm²
Six-Core Opteron 2400	904,000,000	2009	AMD	45 nm	346 mm²
16-Core SPARC T3	1,000,000,000	2010	Sun/Oracle	40 nm	377 mm²
Quad-Core + GPU Core i7	1,160,000,000	2011	Intel	32 nm	216 mm²
Six-Core Core i7	1,170,000,000	2010	Intel	32 nm	240 mm²
8-core POWER7 32M L3	1,200,000,000	2010	IBM	45 nm	567 mm²
8-Core AMD Bulldozer	1,200,000,000	2012	AMD	32nm	315 mm²
Quad-Core + GPU AMD Trinity	1,303,000,000	2012	AMD	32 nm	246 mm²
Quad-core z196	1,400,000,000	2010	IBM	45 nm	512 mm²
Quad-Core + GPU Core i7	1,400,000,000	2012	Intel	22 nm	160 mm²
Dual-Core Itanium 2	1,700,000,000	2006	Intel	90 nm	596 mm²
Six-Core Xeon 7400	1,900,000,000	2008	Intel	45 nm	503 mm²
Quad-Core Itanium Tukwila	2,000,000,000	2010	Intel	65 nm	699 mm²
8-core POWER7+ 80M L3	2,100,000,000	2012	IBM	32 nm	567 mm²
Six-Core Core i7/8-Core Xeon E5	2,270,000,000	2011	Intel	32 nm	434 mm²
8-Core Xeon Nehalem-EX	2,300,000,000	2010	Intel	45 nm	684 mm²
10-Core Xeon Westmere-EX	2,600,000,000	2011	Intel	32 nm	512 mm²
Six-core zEC12	2,750,000,000	2012	IBM	32 nm	597 mm²
8-Core Itanium Poulson	3,100,000,000	2012	Intel	32 nm	544 mm²
62-Core Xeon Phi	5,000,000,000	2012	Intel	22 nm	
Xbox One Main SoC	5,000,000,000	2013	Microsoft		363 mm²

COMPUTER TYPES

Computers are classified based on the parameters like

- Speed of operation
- Cost
- Computational power
- Type of application

Classes of computers

- Personal mobile devices (PMDs)
- Desktop computers
- Servers
- Cluster/warehouse scale computers
- Embedded computers

Computer classes

- **Personal Mobile Devices (PMDs)**
  - Wireless devices with multimedia interface such as smart phones and tablet computers
  - Constraints
    - Cost
    - Energy efficiency
    - Size requirement
- **Desktop computing**
  - Low end note books to high end workstations, battery operated lap tops etc..
  - Constraints
    - Cost
    - performance

Computer classes

- **Servers**
  - Large scale and more reliable file computing services
  - Key features:

  1)Availability,

  2) Scalability,

  3) Throughput e.g. transactions per minute or results per minute, capability of handling number of requests per unit time

Computer classes

- **Clusters/Warehouse scale computers (WSC)**
  - Software as a Service (SaaS) like social networking, search, multiplayer games, video sharing etc has led to *clusters.*
  - Clusters are collections of desktop computers or servers connected by local area networks to act as a single larger computer.
  - Largest clusters are termed as warehouse scale computers
  - Critical factors
    - Price-performance
    - Power
    - Availability just like servers
    - Scalability through LANs
  - WSCs are as expensive as Super computers. But their objectives are different

Computer classes

- **Embedded computers**
  - Found in everyday machines: microwave ovens, washing machines, printers
  - Constraints
    - Cost
    - Performance but not as critical as cost

Feature	Personal mobile device (PMD)	Desktop	Server	Clusters/warehouse-scale computer	Embedded
Price of system	$100–$1000	$300–$2500	$5000–$10,000,000	$100,000–$200,000,000	$10–$100,000
Price of micro-processor	$10–$100	$50–$500	$200–$2000	$50–$250	$0.01–$100
Critical system design issues	Cost, energy, media performance, responsiveness	Price-performance, energy, graphics performance	Throughput, availability, scalability, energy	Price-performance, throughput, energy proportionality	Price, energy, application-specific performance

Basic Terminology

- Computer

– A device that accepts input, processes data, stores data, and produces output, all according to a series of stored instructions.

- Hardware

– Includes the electronic and mechanical devices that process the data; refers to the computer as well as peripheral devices.

Basic Terminology

- Software

– A computer program that tells the computer how to perform particular tasks.

- Peripheral devices

– Used to expand the computer's input, output and storage capabilities.

- Network

– Two or more computers and other devices that are connected, for the purpose of sharing data and programs.

Basic Terminology

- **Input**

Whatever is put into a computer system.

- **Data**

Refers to the symbols that represent facts, objects, or ideas.

- **Information**

The results of the computer storing data as bits and bytes; the words, numbers, sounds, and graphics.

- **Output**

Consists of the processing results produced by a computer.

- **Processing**

Manipulation of the data in many ways.

- **Memory**

Area of the computer that temporarily holds data waiting to be processed, stored, or output.

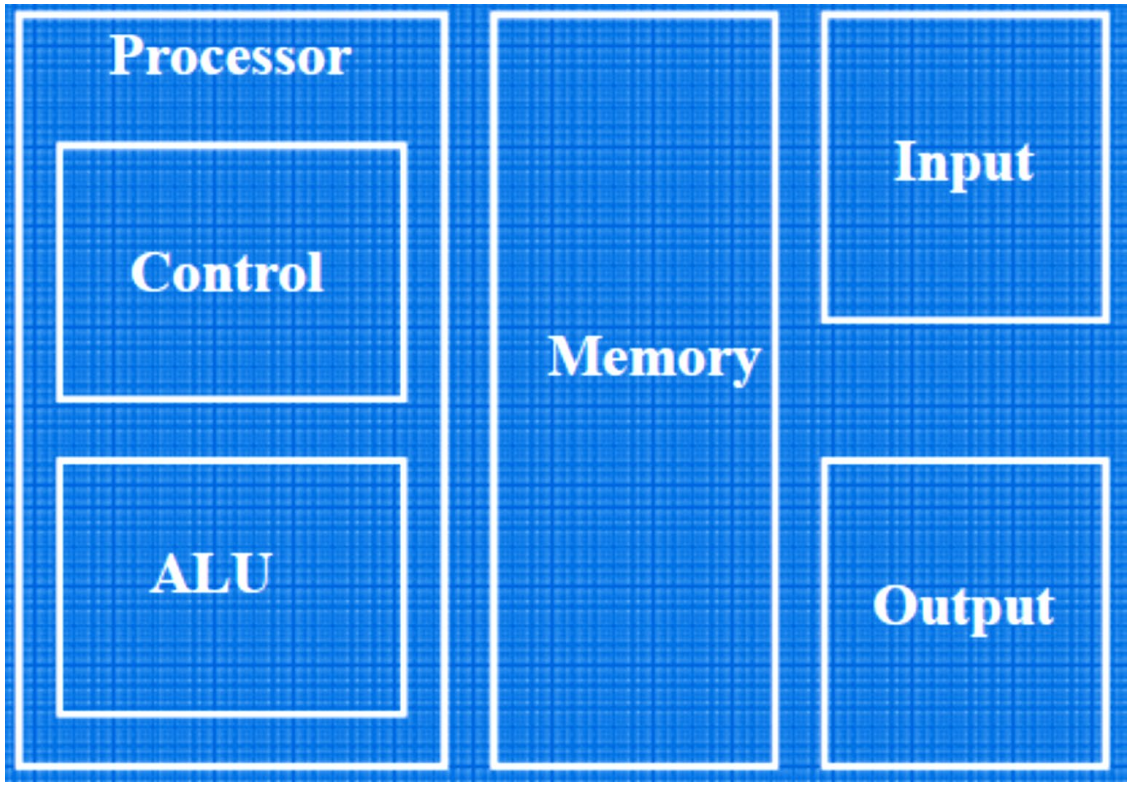
- **Storage**

Area of the computer that holds data on a permanent basis when it is not immediately needed for processing.

Functional Units of Computer

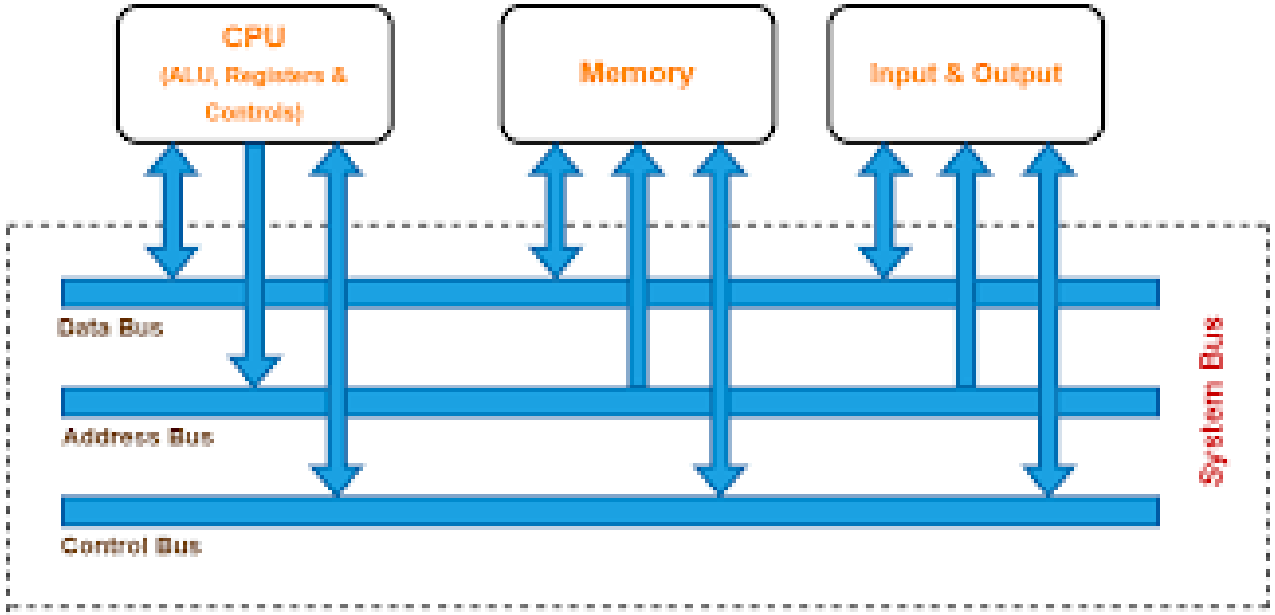
- Input Unit
- Output Unit
- Central processing Unit (ALU and Control Units)
- Memory
- Bus Structure

The Big Picture



Bus System

- The computer system bus is **the method by which data is communicated between all the internal pieces of a computer**. It connects the processor to the RAM, to the hard drive, to the video processor, to the I/O drives, and to all the other components of the computer.



INPUT UNIT:

- Converts the external world data to a binary format, which can be understood by CPU
  - Mouse, Joystick etc

OUTUT UNIT:

- Converts the binary format data to a format that a common man can understand
  - Monitor, Printer, LCD, LED etc

CPU

- The “brain” of the machine
- Responsible for carrying out computational task
- Contains ALU, CU, Registers
- ALU performs Arithmetic and logical operations
- CU provides control signals in accordance with some timings which in turn controls the execution process
- Register stores data and result and speeds up the operation

MEMORY

- Stores data, results, programs
- Two class of storage

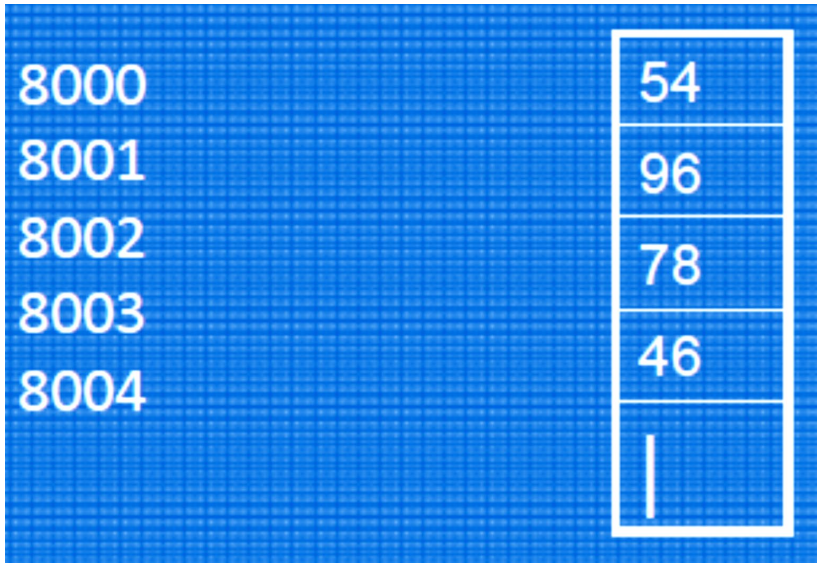
  (i) Primary (ii) Secondary
- Two types are RAM or R/W memory and ROM read only memory
- ROM is used to store data and program which is not going to be changed.
- Secondary storage is used for bulk storage or mass storage

Performance

- Time taken by the system to execute a program
- Parameters which influence the performance are
  - Clock speed
  - Type and number of instructions available
  - Average time required to execute an instruction
  - Memory access time
  - Power dissipation in the system
  - Number of I/O devices and types of I/O devices connected
  - The data transfer capacity of the bus

Assignment of byte addresses

- Little Endian (e.g., in DEC, Intel)
  - low order byte stored at lowest address
    - byte0 byte1 byte2 byte3
    - Eg: 46,78,96,54 (32 bit data)
    - H BYTE ←L BYTE



Assignment of byte addresses

- Big Endian (e.g., in IBM, Motorolla, Sun, HP)
  - high order byte stored at lowest address
  - byte3 byte2 byte1 byte0

- Programmers/protocols should be careful when transferring binary data between Big Endian and Little Endian machines

Memory Operations

- Today, **general-purpose computers** use a set of instructions called a **program** to process data.
- A computer executes the program to create output data from input data
- Both program instructions and data operands are stored in memory
- Two basic operations requires in memory access
  - Load operation (Read or Fetch)-Contents of specified memory location are read by processor
  - Store operation (Write)-Data from the processor is stored in specified memory location

Function

- ALL computer functions are:
  - Data PROCESSING
  - Data STORAGE
  - Data MOVEMENT
  - CONTROL

Data, Information

Coordinates How Information is Used

- NOTHING ELSE!

Classification of Instructions

- BASIC 4 TYPES OF OPERATION:
  - Data transfer between memory and processor register
  - Arithmetic and logic operation
  - Program sequencing and control
  - I/O transfer

Instruction set Architecture

- RISC (Reduced Instruction Set Computer) Architectures
  - Memory accesses are restricted to load and store instruction, and data manipulation instructions are register to register.
  - Addressing modes are limited in number.
  - Instruction formats are all of the same length.
  - Instructions perform elementary operations

- CISC (Complex Instruction Set Computer) Architectures
  - Memory access is directly available to most types of instruction.
  - Addressing mode are substantial in number.
  - Instruction formats are of different lengths.
  - Instructions perform both elementary and complex operations.

Instruction set Architecture

- RISC (Reduced Instruction Set Computer) Architectures
  - Large register file
  - Control unit: simple and hardwired
  - pipelining

- CISC (Complex Instruction Set Computer) Architectures
  - Register file: smaller than in a RISC
  - Control unit: often micro-programmed
  - Current trend
    - CISC operation →a sequence of RISC-like operations

CISC Examples

- Examples of CISC processors are the
  - System/360(excluding the 'scientific' Model 44),
  - VAX,
  - PDP-11,
  - Motorola 68000 family
  - Intel x86 architecture based processors.

Characteristics of RISC Vs CISC processors

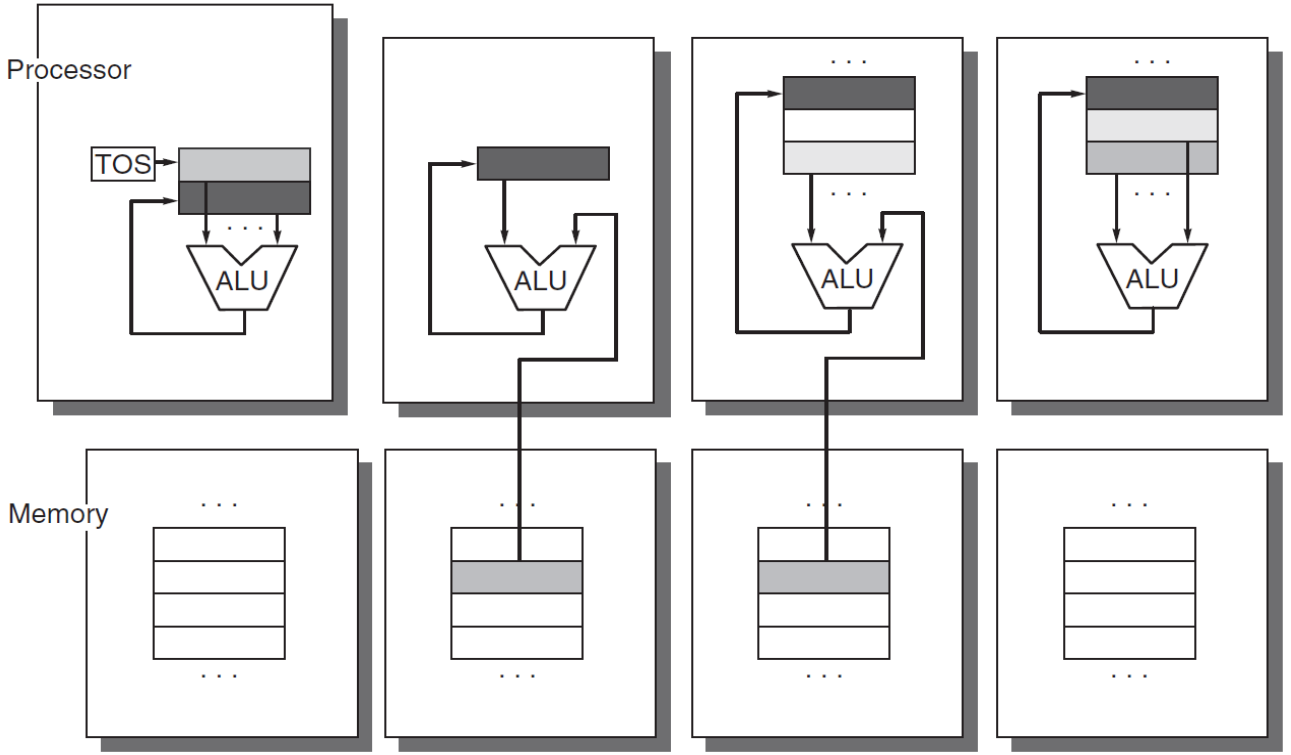
No	RISC	CISC
1	Simple instructions taking one cycle	Complex instructions taking multiple cycles
2	Instructions are executed by hardwired control unit	Instructions are executed by microprogramed control unit
3	Few instructions	Many instructions
4	Fixed format instructions	Variable format instructions
5	Few addressing mode, and most instructions have register to register addressing mode	Many addressing modes
6	Multiple register set	Single register set
7	Highly pipelined	Not pipelined or less pipelined

Instruction Set Architecture (ISA)

- The portion of the computer visible to the programmer or compiler writer.
- A set of all instruction that a processor can do.
- The type of internal storage in a processor is the most basic differentiation:

Classification:

a) Stack Architecture
b) Accumulator Architecture
c) Register-memory Architecture
d) Register-register/load-store Architecture



Stack	Accumulator	Register (register-memory)	Register (load-store)
`Push A`	`Load A`	`Load R1,A`	`Load R1,A`
`Push B`	`Add B`	`Add R3,R1,B`	`Load R2,B`
`Add`	`Store C`	`Store R3,C`	`Add R3,R1,R2`
`Pop C`			`Store R3,C`

Figure A.2 The code sequence for `C = A + B` for four classes of instruction sets.

Number of memory addresses	Maximum number of operands allowed	Type of architecture	Examples
0	3	Load-store	Alpha, ARM, MIPS, PowerPC, SPARC, SuperH, TM32
1	2	Register-memory	IBM 360/370, Intel 80x86, Motorola 68000, TI TMS320C54x
2	2	Memory-memory	VAX (also has three-operand formats)
3	3	Memory-memory	VAX (also has two-operand formats)

Figure A.3 Typical combinations of memory operands and total operands per typical ALU instruction with examples of computers. Computers with no memory reference per ALU instruction are called load-store or register-register computers. Instructions with multiple memory operands per typical ALU instruction are called register-memory or memory-memory, according to whether they have one or more than one memory operand.

Type	Advantages	Disadvantages
Register-register (0, 3)	Simple, fixed-length instruction encoding. Simple code generation model. Instructions take similar numbers of clocks to execute (see Appendix C).	Higher instruction count than architectures with memory references in instructions. More instructions and lower instruction density lead to larger programs.
Register-memory (1, 2)	Data can be accessed without a separate load instruction first. Instruction format tends to be easy to encode and yields good density.	Operands are not equivalent since a source operand in a binary operation is destroyed. Encoding a register number and a memory address in each instruction may restrict the number of registers. Clocks per instruction vary by operand location.
Memory-memory (2, 2) or (3, 3)	Most compact. Doesn't waste registers for temporaries.	Large variation in instruction size, especially for three-operand instructions. In addition, large variation in work per instruction. Memory accesses create memory bottleneck. (Not used today.)

Figure A.4 Advantages and disadvantages of the three most common types of general-purpose register computers. The notation (*m*, *n*) means *m* memory operands and *n* total operands. In general, computers with fewer alternatives simplify the compiler's task since there are fewer decisions for the compiler to make (see Section A.8). Computers with a wide variety of flexible instruction formats reduce the number of bits required to encode the program. The number of registers also affects the instruction size since you need $\log_2$ (number of registers) for each register specifier in an instruction. Thus, doubling the number of registers takes 3 extra bits for a register-register architecture, or about 10% of a 32-bit instruction.

Addressing Modes

- The ways addresses are specified by instructions, called addressing modes.
- The way by which an operand is specified in an instruction as its part.

– **Register**	`add r1, r2`	`r1 <- r1+r2`
– **Immediate**	`add r1, #5`	`r1 <- r1+5`
– **Direct**	`add r1, (0x200)`	`r1 <- r1+M[0x200]`
– **Register indirect**	`add r1, (r2)`	`r1 <- r1+M[r2]`
– **Displacement**	`add r1, 100(r2)`	`r1 <- r1+M[r2+100]`
– **Indexed**	`add r1, (r2+r3)`	`r1 <- r1+M[r2+r3]`
– **Scaled**	`add r1, (r2+r3*4)`	`r1 <- r1+M[r2+r3*4]`
– **Memory indirect**	`add r1, @(r2)`	`r1 <- r1+M[M[r2]]`
– **Auto-increment**	`add r1, (r2)+`	`r1 <- r1+M[r2], r2++`
– **Auto-decrement**	`add r1, -(r2)`	`r2--, r1 <- r1+M[r2]`