4CS015 Lecture -6

CPU ARCHITECTURE

1. Objectives
2. History of Computers
3. History

1.Objectives

- By the end of this session you will be able to:
  - Describe the history of development of the CPU.
  - Explain how cache, pipelines and superscalar operation can improve performance.

1. Objectives
2. History of Computers
3. History

2.History of Computer

Charles Babbage is considered to the father of computer.

- He designed "**Difference Engine**" in 1822
- He designed a ***fully automatic analytical engine*** in 1842 for performing basic arithmetic functions
- His efforts established a number of principles that are fundamental to the design of any digital computer

2. History of Computers
3. History
4. Von Neumann

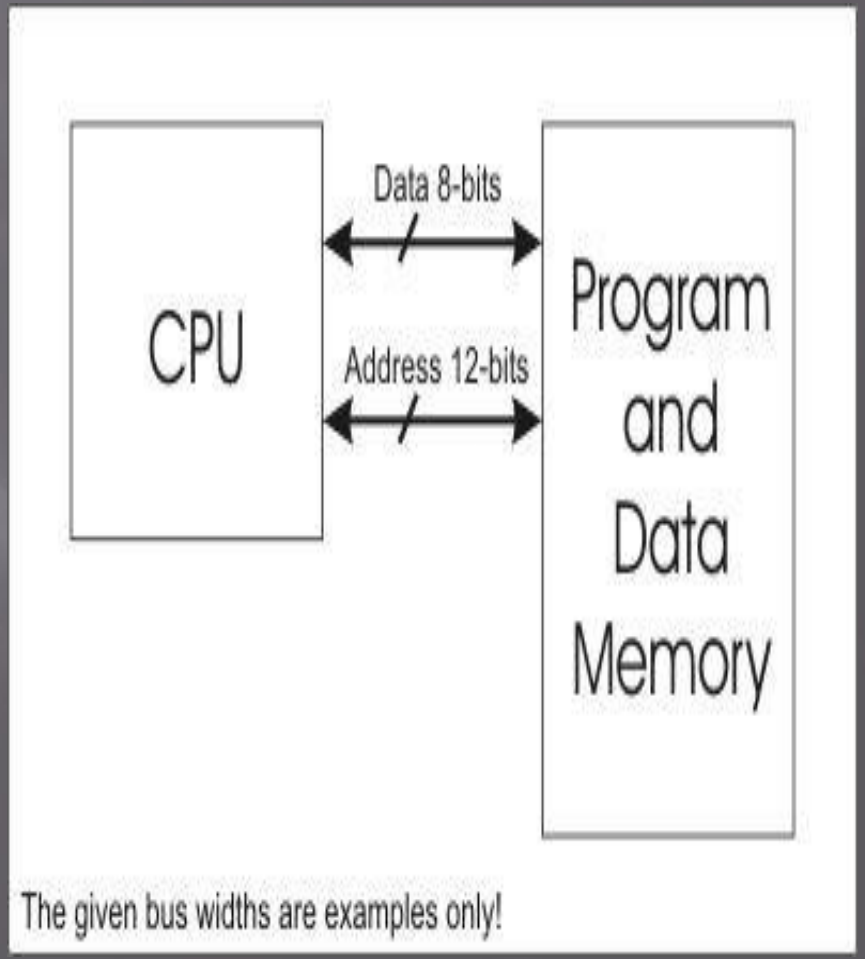
3.History

- First computers programmed by wiring.
- **1944:**
  - EDVAC (**E**lectronic **D**iscrete **V**ariable **A**utomatic **C**omputer) – first conceptual stored program computer.
  - Von Neumann, Eckert, & Mauchly.
- **1948:**
  - First (prototype) stored program computer – Mark I at Manchester University.
  - Harvard Mark III & IV. Howard Aiken.
    - Harvard Architecture / fully automatic / controlled by paper tape / reliable / beginning modern computer era.

3. History
4.Von Neumann
5.Harvard

4.Von Neumann

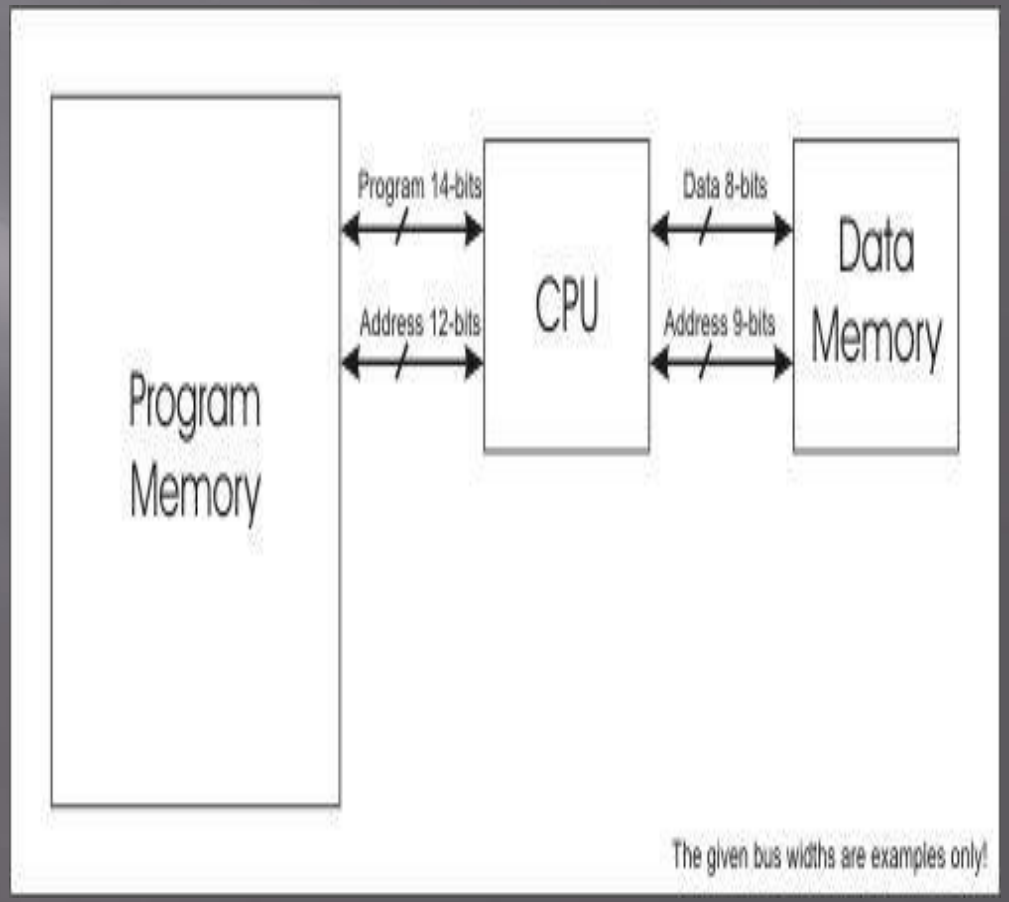
- Classic computer architecture.
- Data and program instructions stored in same memory.
- Fetch-execute cycle.
- Suffers from Von Neumann ‘bottleneck’.



4.Von Neumann
5.Harvard
6. Modified Harvard

5.Harvard

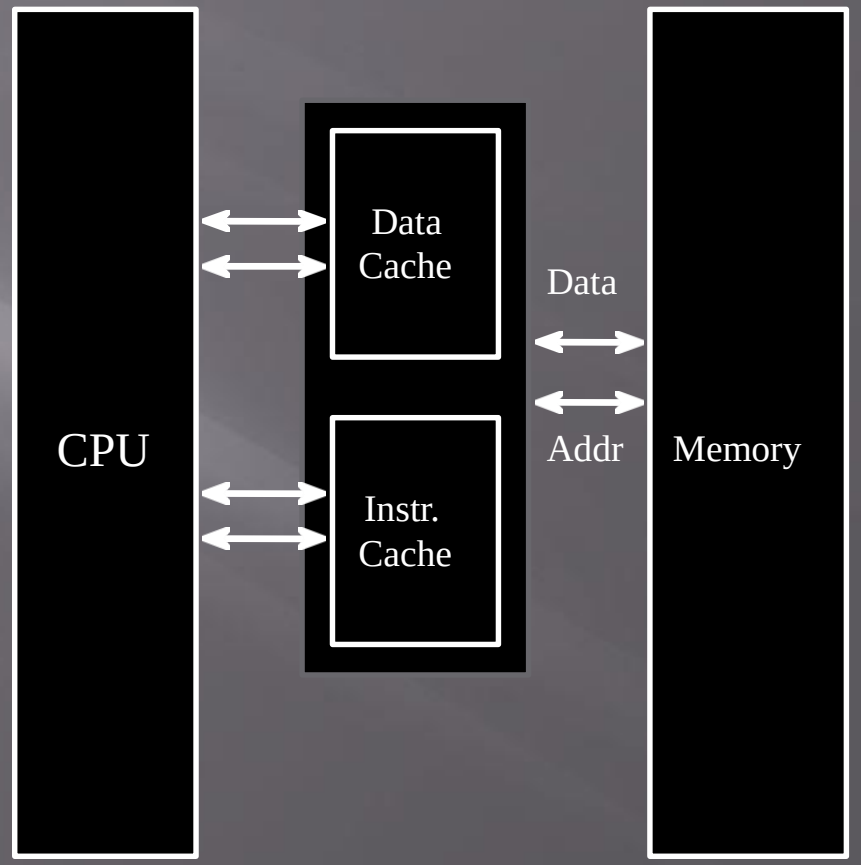
- Data and programs stored in separate memory areas.
- Allows for faster operation.
- Simultaneous access of both data and programs
- Allows data and instruction bus to be diff sizes.



5.Harvard
6. Modified Harvard
7.CPU

6.Modified Harvard

- Hybrid approach – combination of von Neumann and Harvard
- Single Main Memory for both Data and Program
- Separate High-speed Memory Caches for Data and Instructions
- Simultaneous access of both data and programs from caches



7. CPU

5.Harvard
6. Modified Harvard
7.CPU

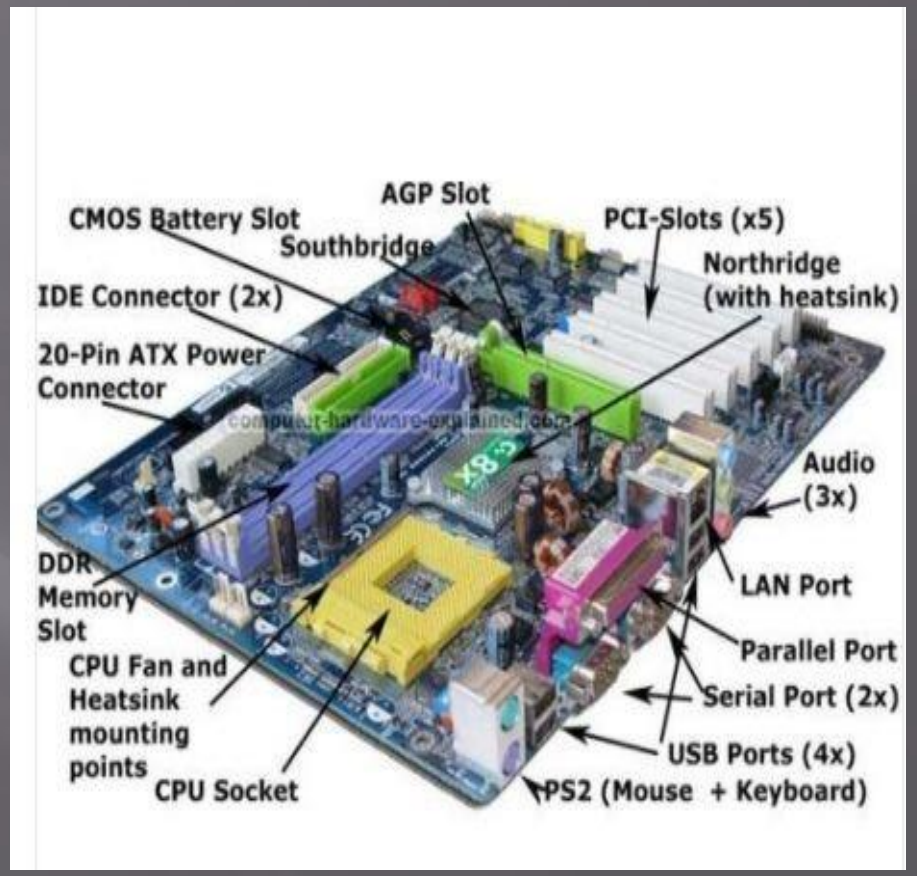
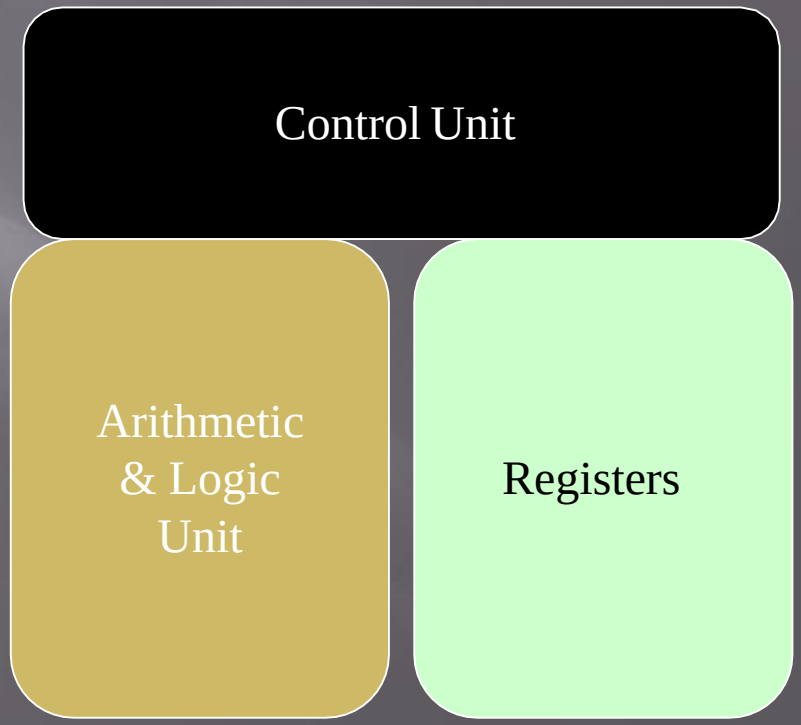


Fig: Mother Board

5.Harvard
6. Modified Harvard
7.CPU

7.CPU

- CPU consists of transistors, combined together as gates.
- CPU works in cycles – fetch, decode, execute.
- Usually each step handled by different part of CPU.
- Where would adders be found?



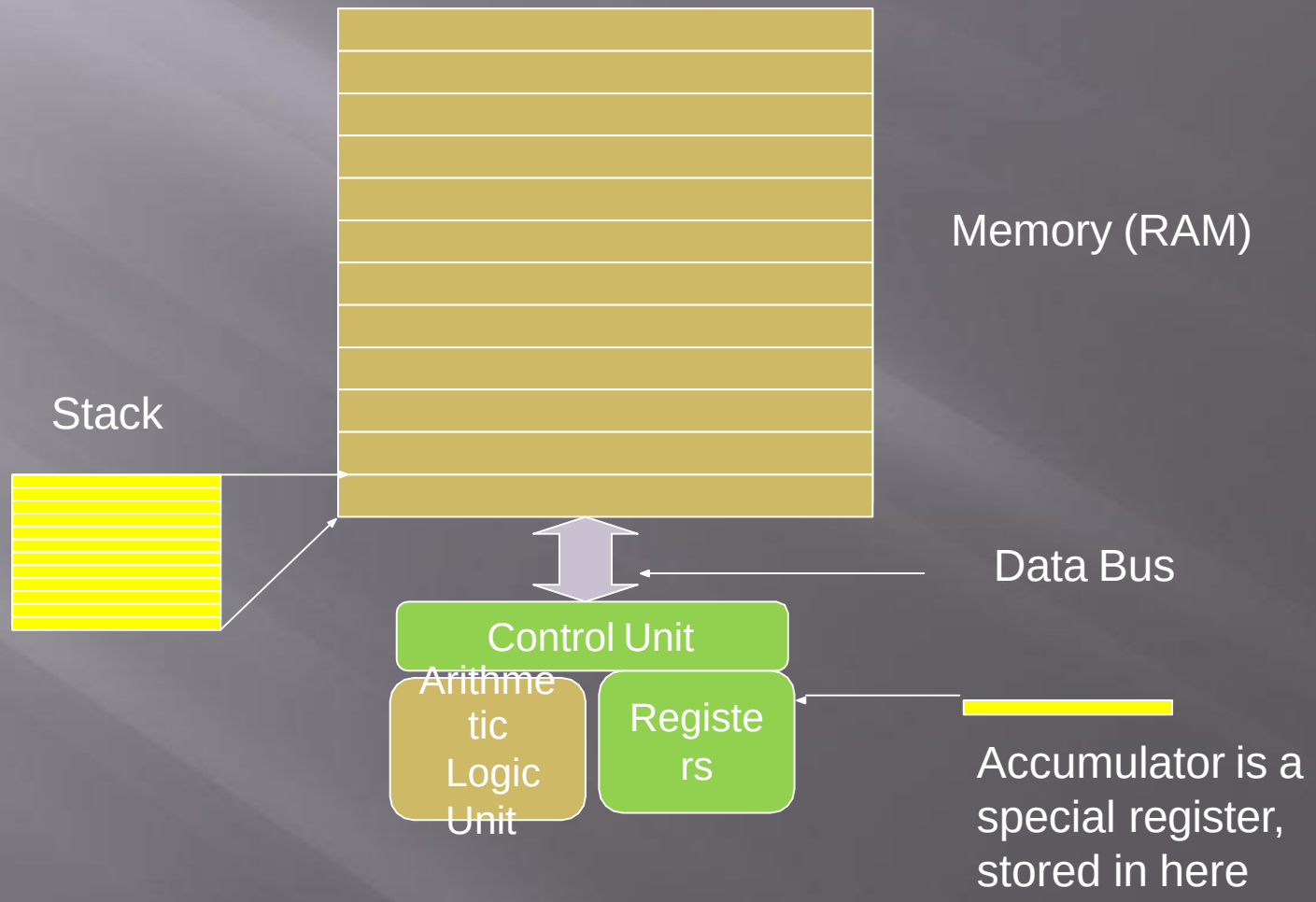
7. CPU
7.1 Types of CPU design
7.2 Example Architecture
7.3 Architecture Bit Sizing

7.1 Types of CPU design

- **Accumulator:**
  - All ALU operations work on data in accumulator (special register).
- **Stack:**
  - All ALU operations work on data stored on the stack.
- **Register-register:**
  - All ALU operations work on data stored in registers.

7. CPU
7.1 Types of CPU design
7.2 Example Architecture
7.3 Architecture Bit Sizing

7.2 Example Architecture



7. CPU
7.1 Types of CPU design
7.2 Example Architecture
7.3 Architecture Bit Sizing

7.3 Architecture Bit Sizing

- Most CPU's are rated by the number of bits they have:
  - 16 bit (8086).
  - 32 bit (80386).
  - 64 bit (Core 2)

8. Bus

- A bus is a high-speed internal connection. Buses are used to send control signals and data between the processor and other components.

Three types of bus are used.

- **Address bus** - carries memory addresses from the processor to other components such as primary storage and input/output devices. The address bus is ***unidirectional.***

- **Data bus** - carries the data between the processor and other components. The data bus is ***bidirectional***.

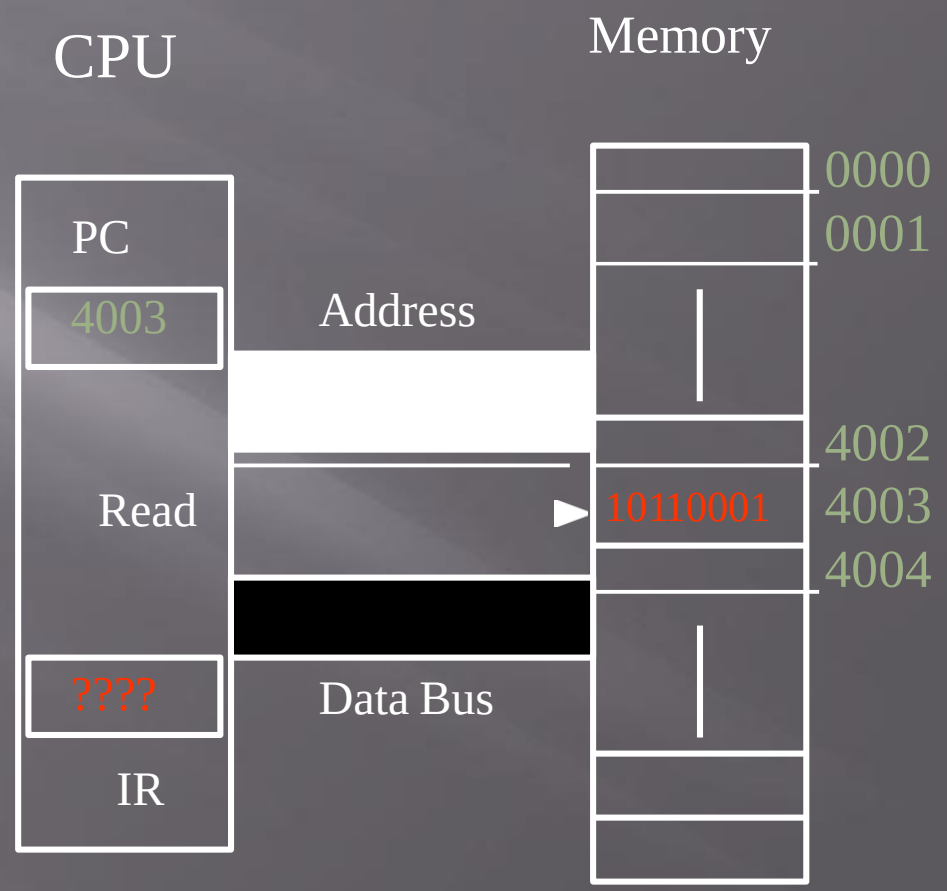
- **Control bus** - carries control signals from the processor to other components. The control bus also carries the clock's pulses. The control bus is ***unidirectional.***

8. Bus
9. Data Bus
10. Register Sizing

9. Data Bus

PC = Program counter. Register / holds the address of the current instruction.

- Determines how much data can be copied from / to memory at a time (cycle?).



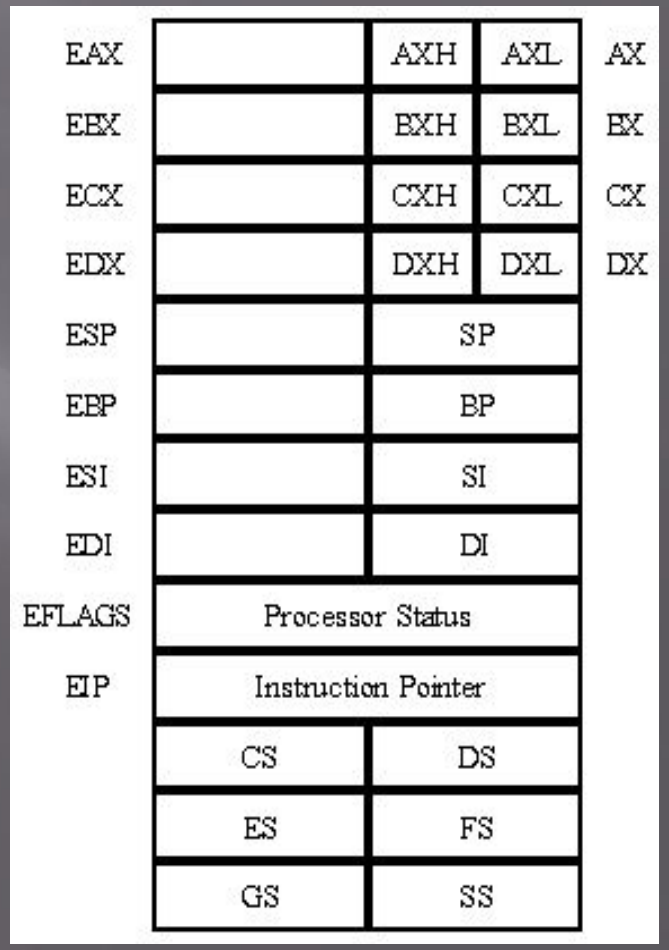
IR = Instruction register. Holds the current instruction while it is executed.

EAX,EBX,ECX,EDX: general purpose registers.
ESP: stack pointer (top).
EBP: Stack base Pointer.
EFLAGS: condition codes.
EIP: program counter (instruction pointer).

10.Register Sizing

9. Data Bus
10. Register Sizing
11. Address Bus

- Registers are the internal ‘memory’ of a CPU.
- They determine the maximum memory that can be addressed (PC).
- They determine the size of ALU operations.
- Can be 8-bit, 16-bit, 32-bit, 64-bt (or more!)

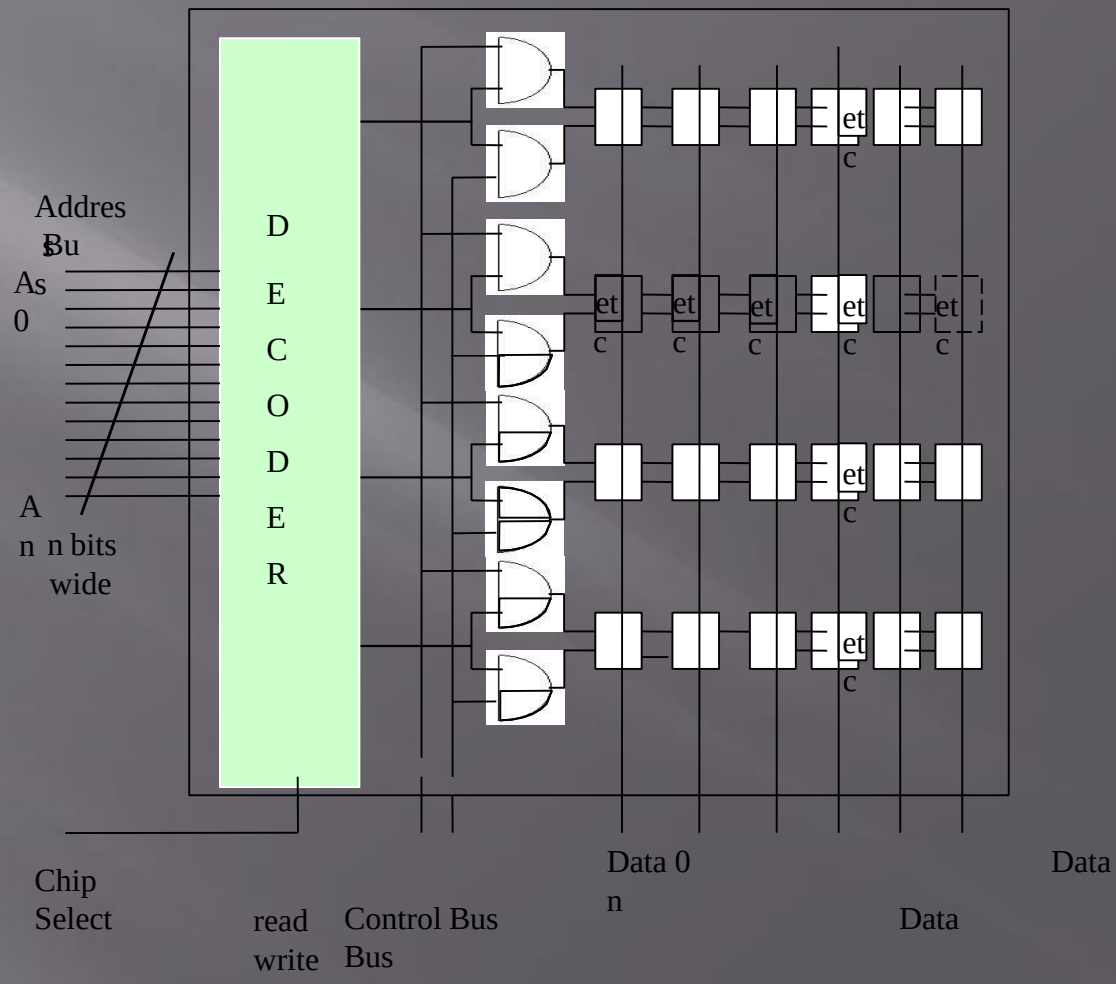


11. Address Bus

10. Register Sizing
11. Address Bus
12.Next Steps

Determines how much memory can be accessed.

Matched with program counter / pointer



10. Register Sizing
11. Address Bus
12.Next Steps

11. Address Bus

- Each ‘line’ in memory corresponds to a bit in a register (the MAR – memory address register).
- So 8 lines =8 bits =2^{8} =??
- 32 lines =32 bits =2^{32} =4GB.
- 64 lines =64 bits =2^{64} =18.5 EB (exabytes), 1.85×10^{19}bytes

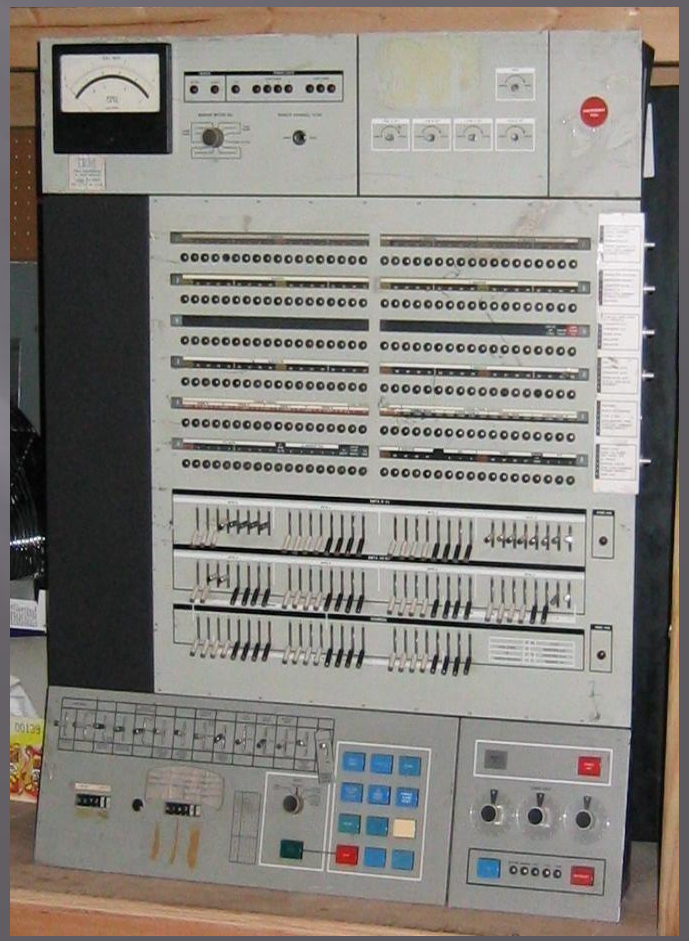
11. Address Bus
12.Next Steps
13. System 360

12. Next Steps…

- Late fifties / early sixties number computers available increased rapidly.
- Generally each generation had a different architecture than previous one (enhancements / improvements).
  - Different instruction sets.
  - Machines customised for customers.
- Was this a problem?
- IBM solution: system 360.

13. System 360

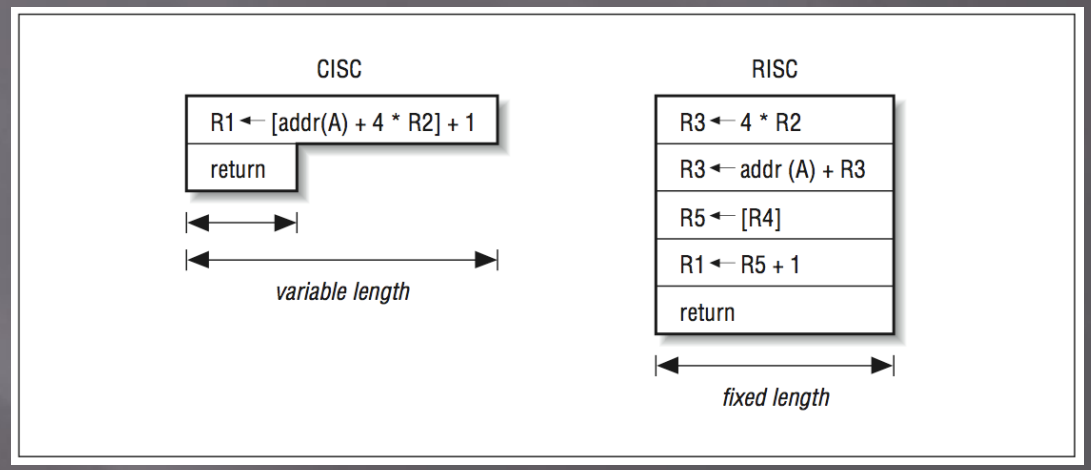
- System 360 popularised use of microcode across the ‘range’.
  - **i.e.** same instruction set.
  - **Example:** Multiplication.
- Any program written for one model would work on all.
- Binary compatibility.



13. System 360
14. CISC
15.Evolution

14. CISC - Complex Instruction Set Computer

- Originally, CPU speed and Memory speed the same.
- Instructions varied in size, 1 – 5 words (16 bit).
- Depending on data bus size, could take up to 5 cycles / instruction.
- Decoding may include microcode.
- **CISC** – designed to make programming easier – either for assembly programmer or compiler programmer.



14. CISC
15.Evolution
16. Instruction Usage

15. Evolution

- Mid seventies – John Cocke @ IBM initiated research on performance of microcoded CPU's.
- 1980 – Patterson built on this work, coined term RISC.
- **Conclusions:**
  - Most programs use only small % of available instructions.
  - Microcoded instructions not always best / most efficient way of performing task.

15.Evolution
16. Instruction Usage
17. RISC

16. Instruction Usage

Instruction type	Usage X86	Usage (ARM)
Data movement	38%	43%
Control flow	22%	23%
Arithmetic operations	14%	15%
Comparisons	16%	13%
Logical operations	6%	5%
Other	4%	1%

- Which is the most important type of instruction to optimise?

16. Instruction Usage
17. RISC
18. RISC v. CISC

17. RISC - Reduced instruction set computer

- RISC should really be Reduced Complexity Instructions (RCISC)
- Ideal specification:
  - Each instruction executes in one cycle – increases processing efficiency.
  - No microcode.
  - All instructions work on registers, except load / store.
  - May need more store as have more instructions.

17. RISC
18. RISC v. CISC
19. Example: VAX Vs. MIPS

18. RISC v. CISC

- **Performance:**
- RISC optimises cycles / instruction.
- CISC optimises instructions per program.
- Trade – off.
- How do we optimise cycle time?

18. RISC v. CISC
19. Example: VAX Vs. MIPS
20. Modern Enhancements

19.Example: VAX Vs. MIPS

Description	Value (mean)
VAX CPI	9.9
MIPS CPI	1.7
CPI Ratio (VAX/MIPS)	5.8
Instruction Ratio (MIPS/VAX)	2.2
RISC factor	2.7

- **Cycles Per Instruction (CPI):** instructions may take one or more cycles to complete.
- **VAX** – CISC architecture. **MIPS** – RISC architecture.
- RISC factor = CPI ratio / Instruction ratio.

19. Example: VAX Vs. MIPS
20. Modern Enhancements
21. Clock Speed

20. Modern Enhancements

- How do we improve cycle times?
- Cycle time = **I-time** + **E-time** where
  - **I-time** = time taken to fetch instruction from memory (or cache).
  - **E-time** – time taken by ALU to execute instruction.

21. Clock Speed

- System clock like ‘heartbeat’ for system.
- All system synchronised to clock.
- Ideal condition is CPU performs one instruction (or more!) per clock cycle.

21. Clock Speed
22. Clock Cycles
23. Operations(Original CISC)

22.Clock Cycles

- Cycles measured in Hertz.
- 1 MHz =1 million cycles/sec.
- 1 GHz = 1,000 million cycles/sec.

- Remember ideal: >1 instruction/cycle!

23. Operations (Original CISC)

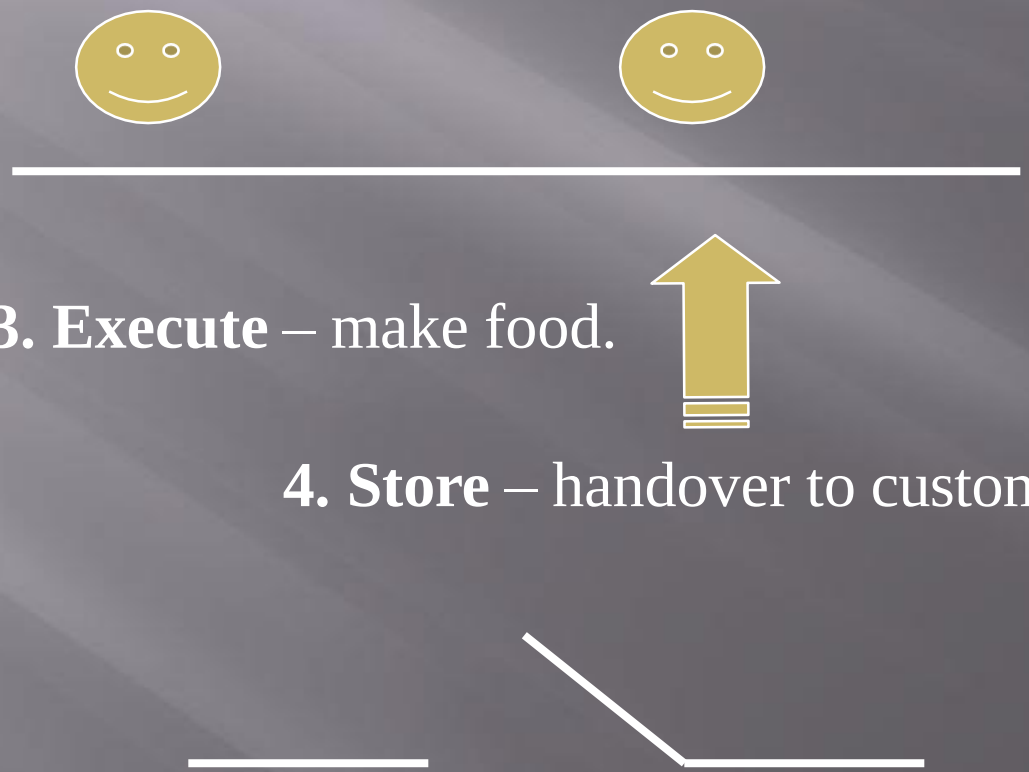
- **Analogy:** The fast food outlet…

1. Fetch – press button, open door.

2. Decode – take order.

24.Operations

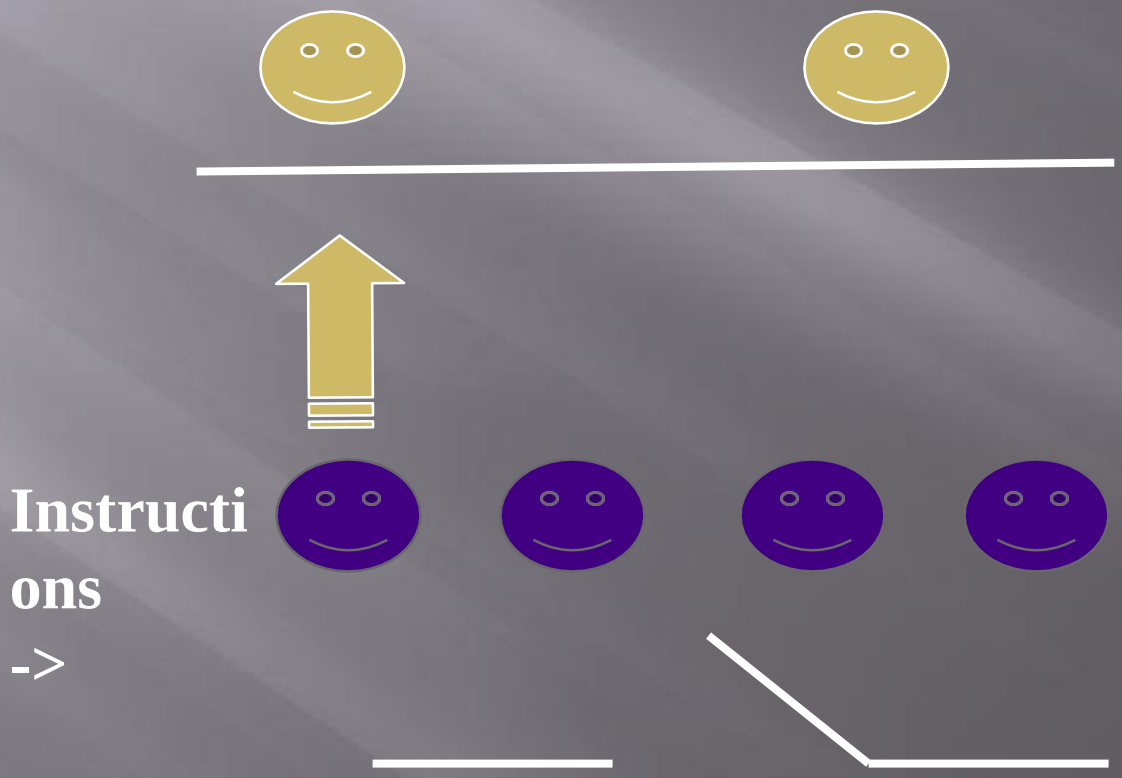
23. Operations (Original CISC)
24.Operations
25. Cache



3. Execute – make food.

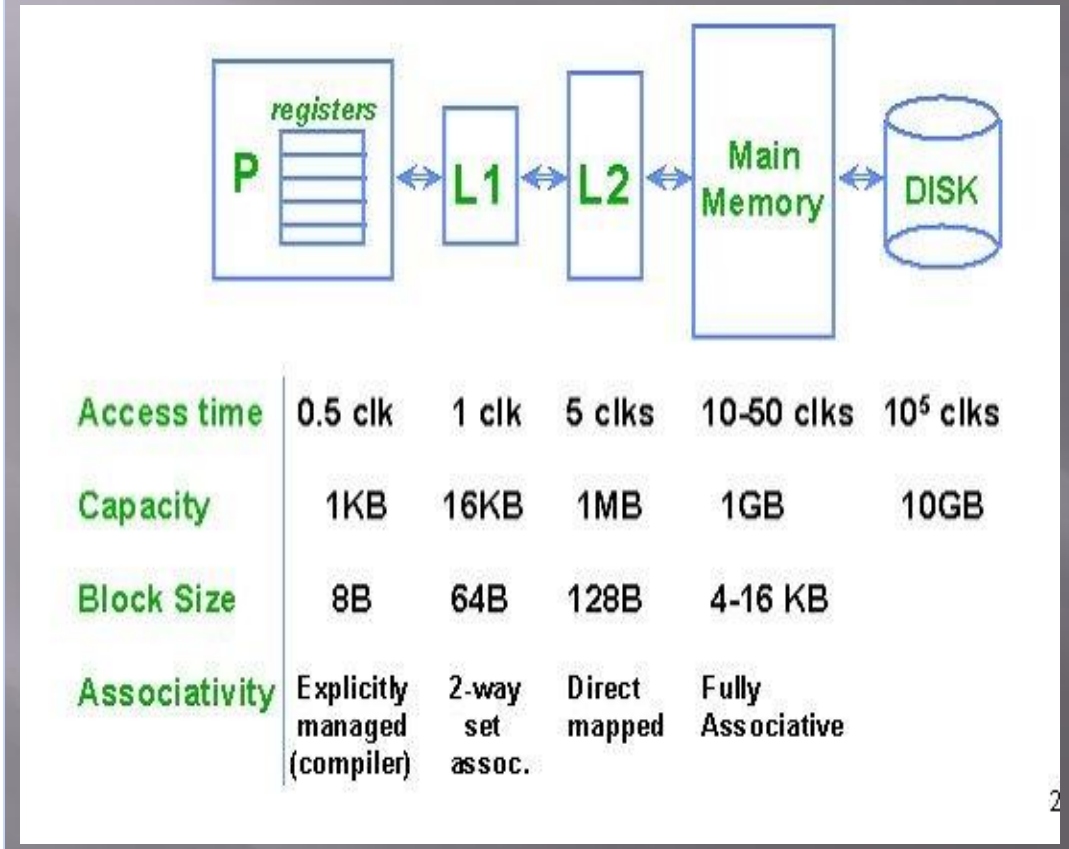
4. Store – handover to customer.

25.Cache

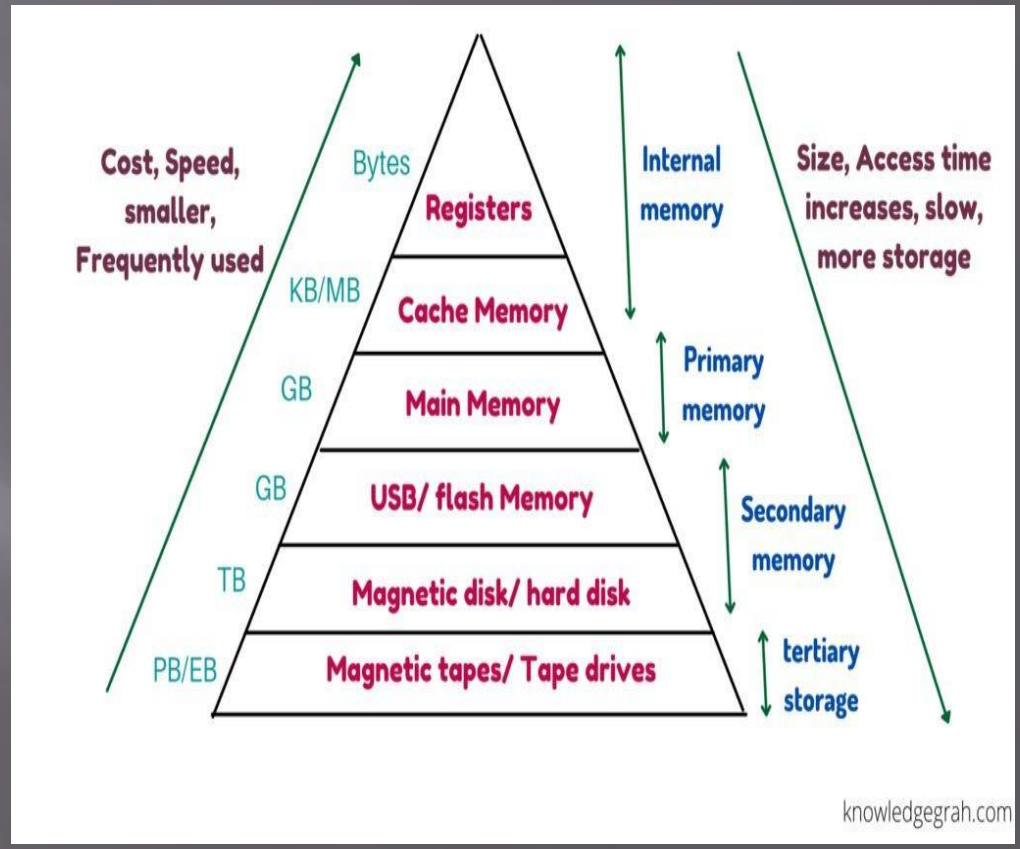


- Expensive and high speed memory.
- Speed up memory retrieval.
- Relatively small amount.

25. Cache

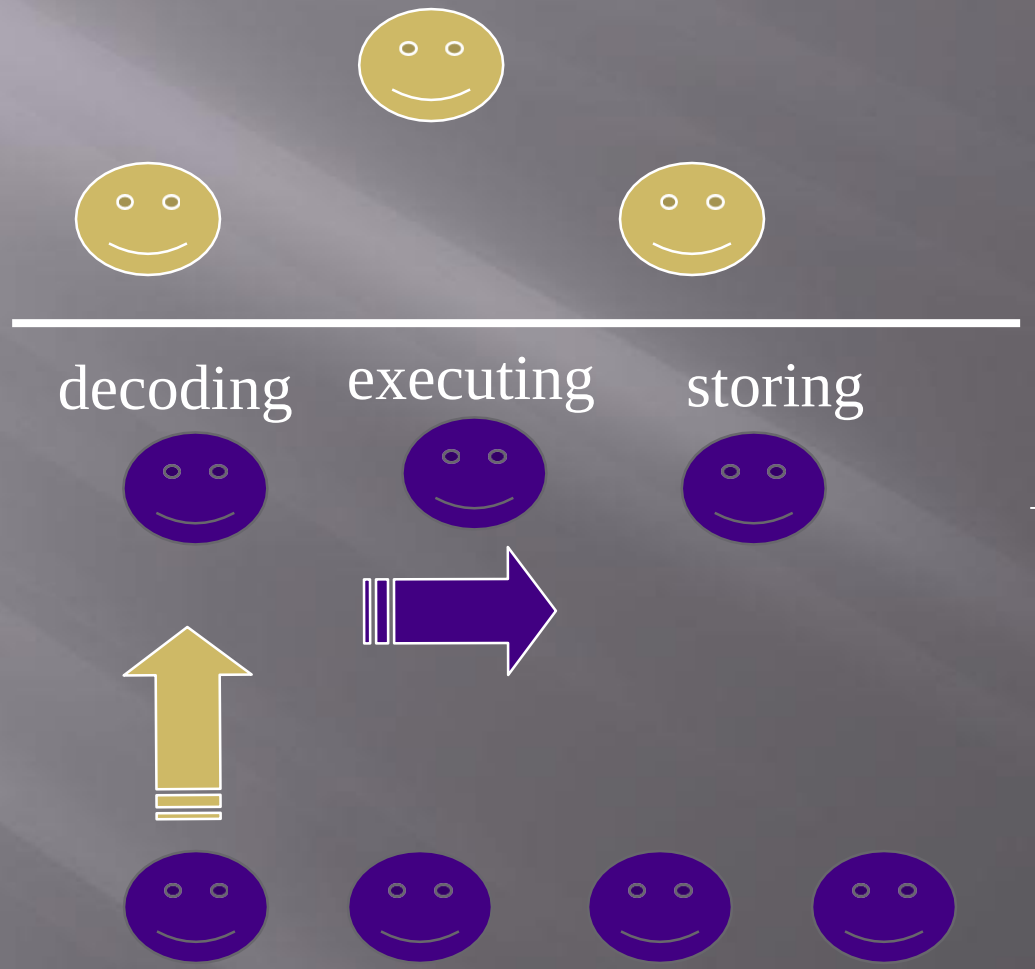


Access time	0.5 clk	1 clk	5 clks	10-50 clks	10^5 clks
Capacity	1KB	16KB	1MB	1GB	10GB
Block Size	8B	64B	128B	4-16 KB	
Associativity	Explicitly managed (compiler)	2-way set assoc.	Direct mapped	Fully Associative	



25. Cache
26. Pipeline
27. Superscalar

26. Pipeline



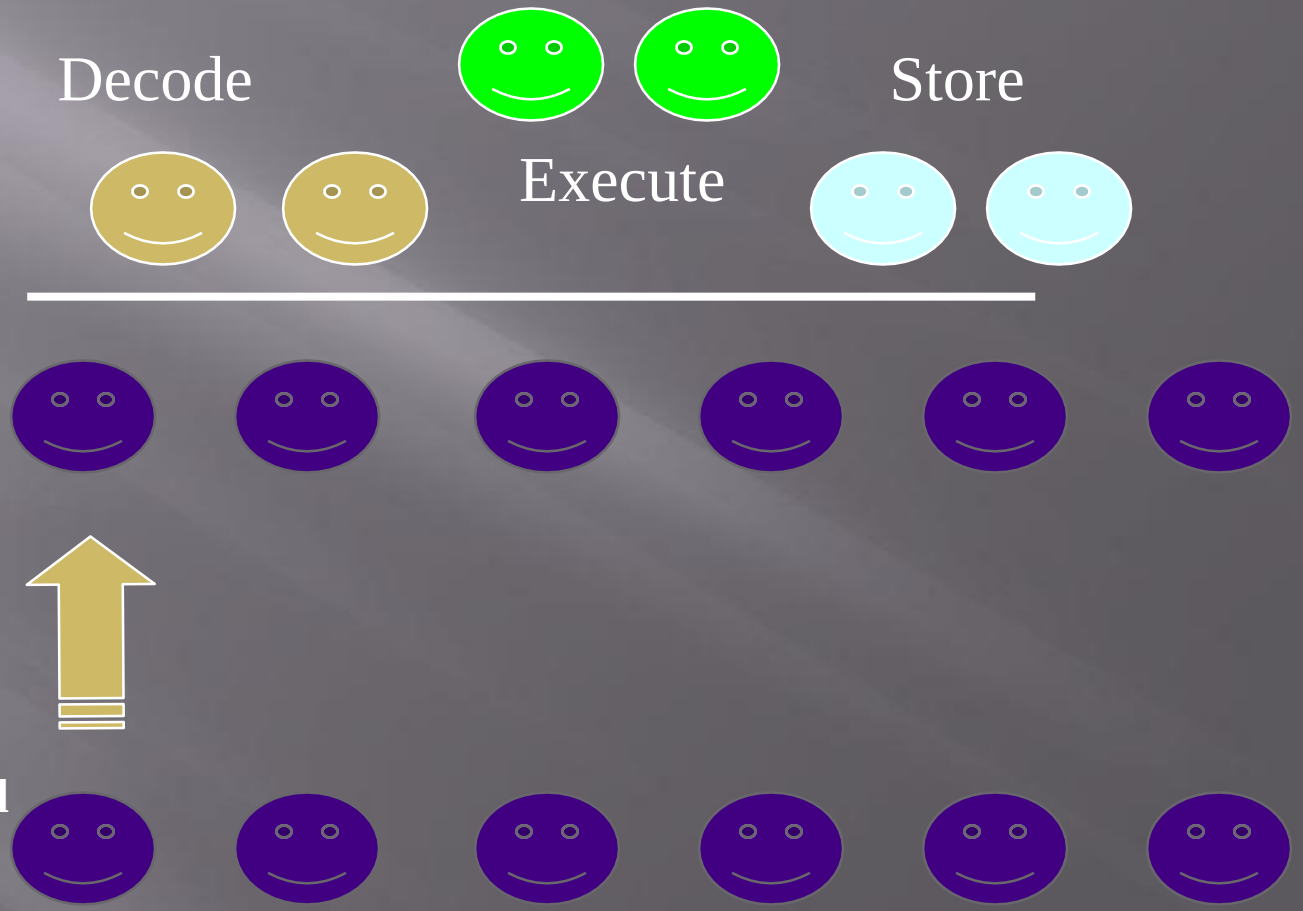
simultaneously

25. Cache
26. Pipeline
27. Superscalar

26. Pipeline

- Concept of using each functional unit simultaneously.
- Theoretically:
  - N stage pipeline =n* increase in speed.
  - Problems in practice? **Pipeline Hazards.**
  - **Instruction 2** operand is the output from **Instruction 1**.
    - Stall / read-after-write hazard.
  - Branching.

27. Superscalar



Multiple, independent, functional units.

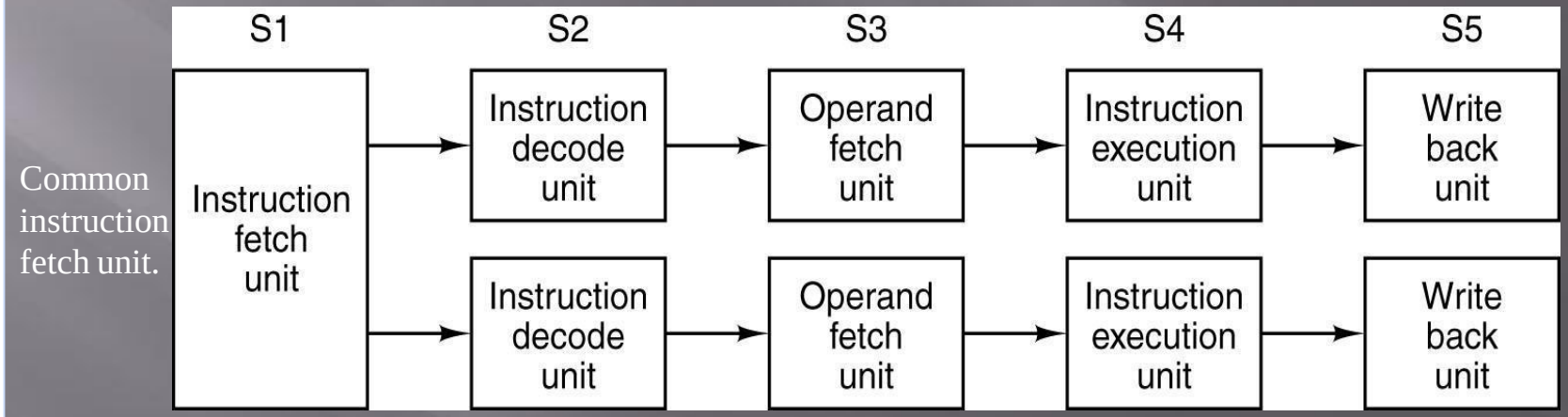
Instruction-level parallelism.

27. Superscalar

Which instructions can be perform simultaneously?

E = a + b F = c + d G = e * f

- Concept of adding more functional units.
- Theoretically:
  - N units = N instructions processing in parallel.
  - Problems in practice? **Order of instruction?**



28. Pentium pro
29. Pentium 4
30. Summary

30.Summary

- Storage options (stack, accumulator or register).
- CISC vs. RISC.
- Modern enhancements (cache, pipelines, superscalar).
- Modern processor examples.