Computer Architecture

Prof. Dr. Nizamettin AYDIN

naydin@yildiz.edu.tr

http://www.yildiz.edu.tr/~naydin

1

The Von Neumann Model/Architecture

- Also called stored program computer (instructions in memory).
- Two key properties:
  - Stored program
    - Instructions stored in a linear memory array
    - Memory is unified between instructions and data
      - The interpretation of a stored value depends on the control signals
        - When is a value interpreted as an instruction?
  - Sequential instruction processing
    - One instruction processed (fetched, executed, and completed) at a time

2

A computing System (Reminder)

- What is a computer?
  - in terms of what?
    - Functional
    - Structural

Functional definition

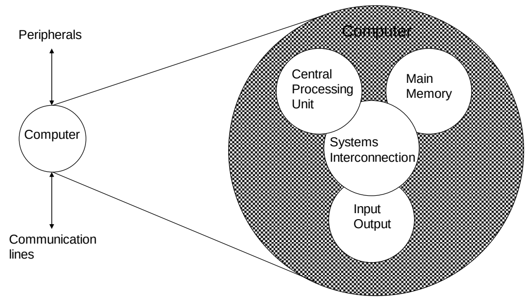
- Data processing
- Data storage
- Data movement
- Control

Structural definition

- Central processing unit
- Main memory
- Input/Output
- System interconnection

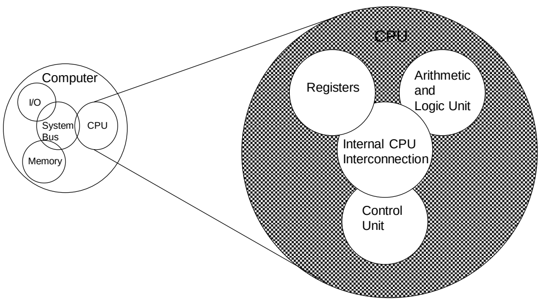
3

Structure - Top Level



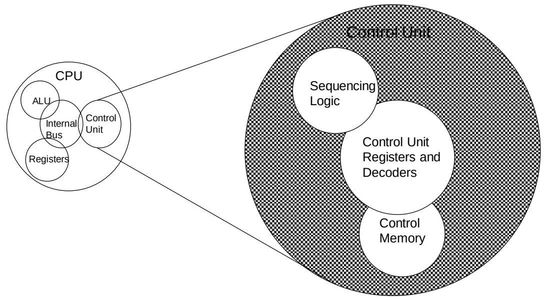
4

Structure - The CPU



5

Structure - The Control Unit



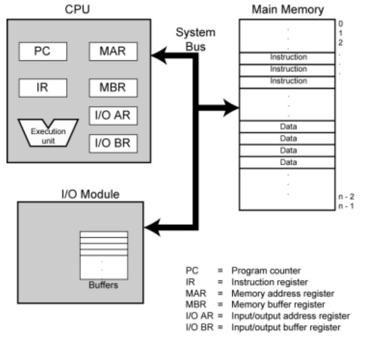
6

The Stored Program Computer

- 1943: ENIAC
  - Presper Eckert and John Mauchly -- first general electronic computer. **(or was it John V. Atanasoff in 1939?)**
  - Hard-wired program -- settings of dials and switches.
- 1944: Beginnings of EDVAC
  - among other improvements, includes program stored in memory
- 1945: John von Neumann
  - wrote a report on the stored program concept, known as the *First Draft of a Report on EDVAC*
- The basic structure proposed in the draft became known as the “von Neumann machine” (or model).
  - a *memory*, containing instructions and data
  - a *processing unit*, for performing arithmetic and logical operations
  - a *control unit*, for interpreting instructions

7

Von Neumann Model



8

ENIAC - details

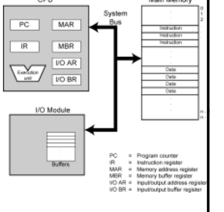
http://www.seas.upenn.edu/~museum/

- Decimal (not binary)
- 20 accumulators of 10 digits
- Programmed manually by switches
- 18,000 vacuum tubes
- 30 tons
- 15,000 square feet
- 140 kW power consumption
- 5,000 additions per second

9

Interface to Memory

- How does processing unit get data to/from memory?
- MAR: Memory Address Register
- MBR (MDR): Memory Buffer (Data) Register



To LOAD a memory location (A):

1. Write the address (A) into the MAR.
2. Send a “read” signal to the memory.
3. Read the data from MBR.

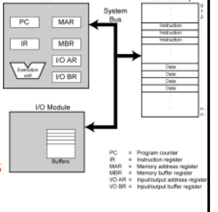
To STORE a value (X) to a location (A):

1. Write the data (X) to the MBR.
2. Write the address (A) into the MAR.
3. Send a “write” signal to the memory.

10

Processing Unit

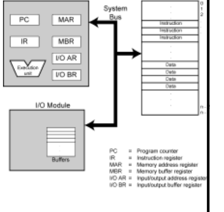
- Functional Units (Execution unit)
  - ALU = Arithmetic and Logic Unit
  - could have many functional units. some of them special-purpose (multiply, square root, …)
- Registers
  - Small, temporary storage
  - Operands and results of functional units
- Word Size
  - number of bits normally processed by ALU in one instruction
  - also width of registers



11

Input and Output

- Devices for getting data into and out of computer memory
- Each device has its own interface, usually a set of registers (I/OAR and I/OBR)
- Some devices provide both input and output
  - disk, network
- Program that controls access to a device is usually called a *driver*.



12

Control Unit

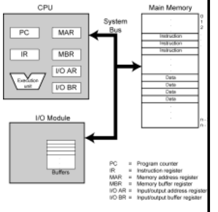
•Orchestrates execution of the program

•Instruction Register (IR) contains the *current instruction*.

•Program Counter (PC) contains the *address* of the next instruction to be executed.

•Control unit:

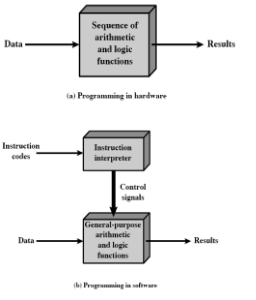
– reads an instruction from memory
  • the instruction's address is in the PC
– interprets the instruction, generating signals that tell the other components what to do
  • an instruction may take many *machine cycles* to complete



13

Program Concept

- Hardwired systems are inflexible
- General purpose hardware can do different tasks, given correct control signals
- Instead of re-wiring, supply a new set of control signals



14

What is a program?

- A sequence of steps
- For each step, an arithmetic or logical operation is done
- For each operation, a different set of control signals is needed

15

Instruction Processing

Fetch instruction from memory

Decode instruction

Evaluate Address

fetch OPerands from memory

EXecute operation

Store result

16

Instruction

•The instruction is the fundamental unit of work.

•Specifies two things:

– opcode: operation to be performed
– operands: data/locations to be used for operation

•An instruction is encoded as a sequence of bits.
(*Just like data!*)

– **Often, but not always, instructions have a fixed length, such as 16 or 32 bits.**
– **Control unit interprets instruction: generates sequence of control signals to carry out operation.**
– **Operation is either executed completely, or not at all.**

•A computer's instructions and their formats is known as its ***Instruction Set Architecture (ISA)***.

17

Instruction Processing: FETCH

•Load next instruction (at address stored in PC) from memory
into Instruction Register (IR).

– Copy contents of PC into MAR.
– Send "read" signal to memory.
– Copy contents of MBR into IR.

•Then increment PC, so that it points to the next instruction in sequence.

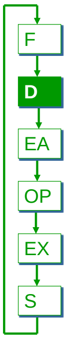
– PC becomes PC+1.

F → D → EA → OP → EX → S

18

Instruction Processing: DECODE

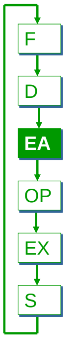
- First identify the opcode.
  - A n-to-2^n decoder asserts a control line corresponding to the desired opcode.
- Depending on opcode, identify other operands from the remaining bits.



19

Instruction Processing: EVALUATE ADDRESS

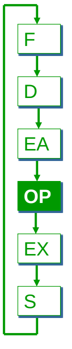
- For instructions that require memory access, compute address used for access.
- Examples:
  - add offset to base register
  - add offset to PC
  - add offset to zero



20

Instruction Processing: FETCH OPERANDS

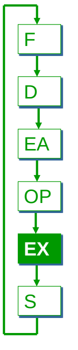
- Obtain source operands needed to perform operation.
- Examples:
  - load data from memory (LDA)
  - read data from register file (ADD)



21

Instruction Processing: EXECUTE

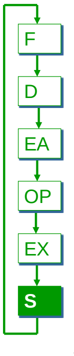
- Perform the operation, using the source operands.
- Examples:
  - send operands to ALU and assert ADD signal
  - do nothing (e.g., for loads and stores)



22

Instruction Processing: STORE RESULT

- Write results to destination. (register or memory)
- Examples:
  - result of ADD is placed in destination register
  - result of memory load is placed in destination register
  - for store instruction, data is stored to memory
    - write address to MAR, data to MBR
    - assert WRITE signal to memory



23

Changing the Sequence of Instructions

- In the FETCH phase, we increment the Program Counter by 1.
- What if we don't want to always execute the instruction that follows this one?
  - examples: loop, if-then, function call
- Need special instructions that change the contents of the PC.
- These are called *control instructions*.
  - jumps are unconditional -- they always change the PC
  - branches are conditional -- they change the PC only if some condition is true (e.g., the result of an ADD is zero)

24

Instruction Processing Summary

•Instructions look just like data -- it's all interpretation.

•Four basic kinds of instructions:
- Data processing instructions
  - Arithmetic and logic instructions (ADD, AND, …)
- Data storage instructions
  - Memory instructions (LDA, STA)
- Data movement instructions
  - I/O instructions (IN, OUT, …)
- Program flow control instructions
  - Test and branch instructions (JMP, BRP, …)

•Six basic phases of instruction processing:

F → D → EA → OP → EX → S

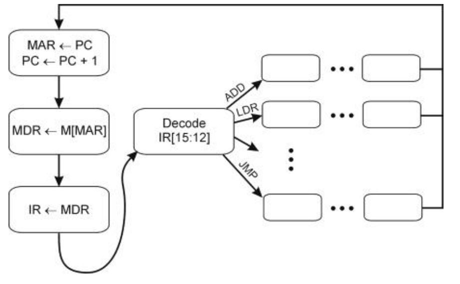
- not all phases are needed by every instruction
- phases may take variable number of machine cycles

25

Control Unit State Diagram

•The control unit is a state machine.

•Here is part of a simplified state diagram:

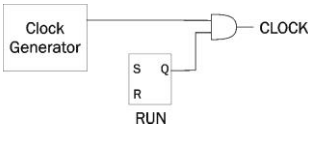


26

Stopping the Clock

•Control unit will repeat instruction processing sequence as long as clock is running.
- **If not processing instructions from your application, then it is processing instructions from the Operating System (OS).**
- **The OS is a special program that manages processor and other resources.**

•To stop the computer:
- **AND the clock generator signal with ZERO**
- **When control unit stops seeing the CLOCK signal, it stops processing.**



27

Dataflow Model of a Computer

Von Neumann model
- An instruction is fetched and executed in control flow order
  - As specified by the instruction pointer
  - Sequential unless explicit control flow instructio

Dataflow model
- An instruction is fetched and executed in data flow order
  - i.e., when its operands are ready
  - i.e., there is no instruction pointer
  - Instruction ordering specified by data flow dependence
    - Each instruction specifies "who" should receive the result
    - An instruction can "fire" whenever all operands are received
  - Potentially many instructions can execute at the same time
  - Inherently more parallel

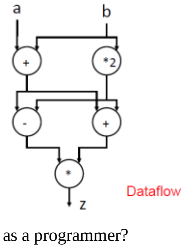
28

von Neumann vs Dataflow

- Consider a von Neumann program (sequential) vs dataflow execution

```
v <= a + b;
w <= b * 2;
x <= v - w
y <= v + w
z <= x * y
```

Sequential



Dataflow

- Which model is more natural to you as a programmer?
- All major instruction set architectures today use von Neumann
  - x86, ARM, MIPS, SPARC, Alpha, POWER PC

29