

CSE 220 Computer Organization

Lecture 6

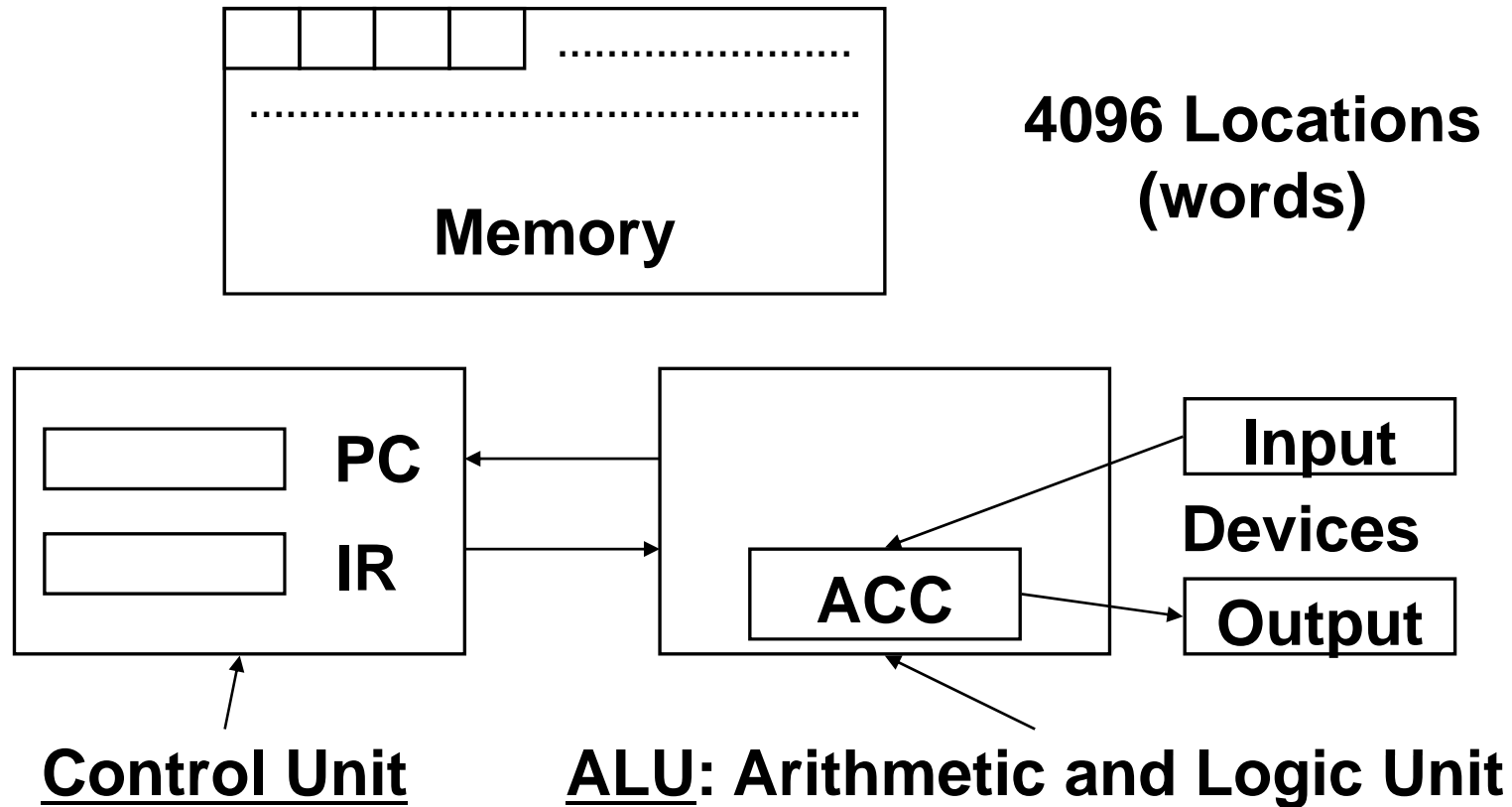
Hussein Badr
Computer Science, Stony Brook University

<http://www.cs.sunysb.edu/~cse220>

Background Preparation

- **Stored program computer**
- **Bits, bytes and words**
- **Radix number system**
- **Binary arithmetic**
- **Floating point numbers and their arithmetic**

John von Neumann Machine (1/7)

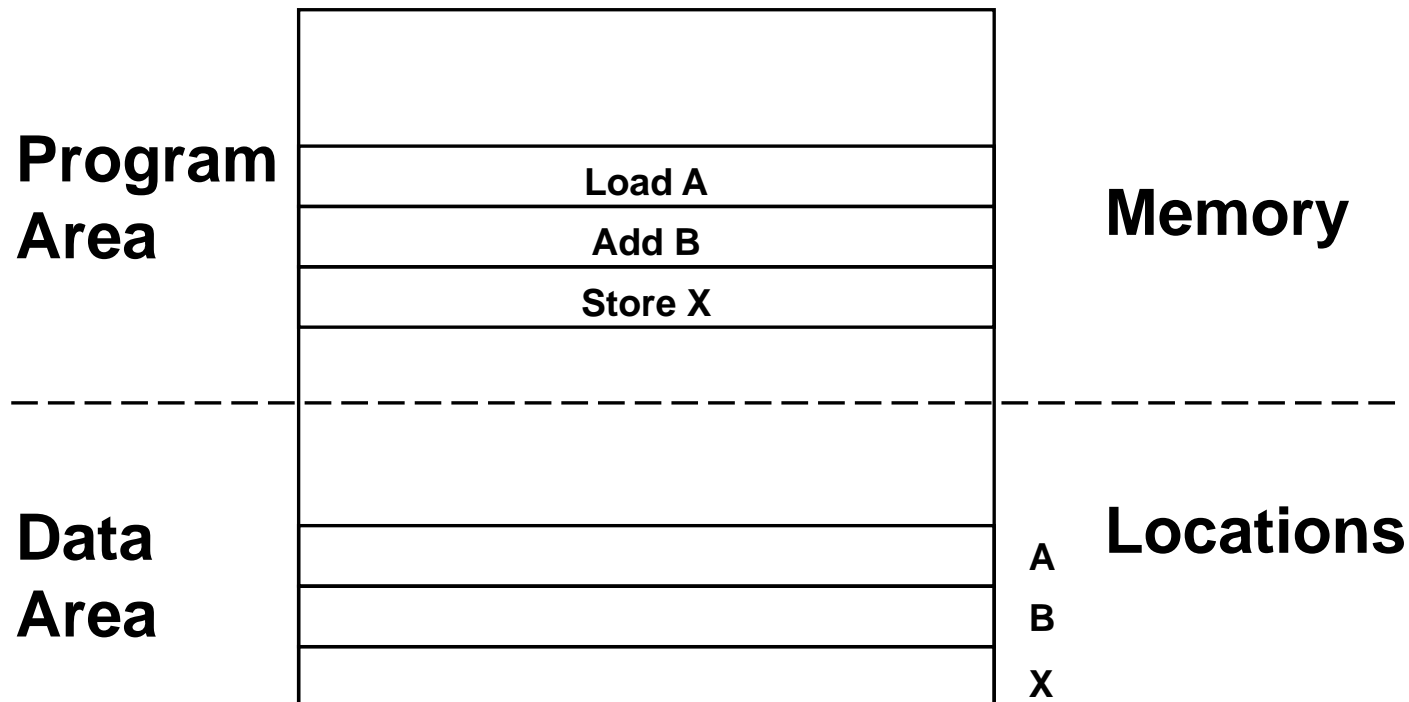


PC → Program Counter
IR → Instruction Register

John von Neumann Machine (2/7)

- For example, a high level language statement $X=A+B$ would be translated as follows.

Load A	Copy A to accumulator
Add B	Add contents of B
Store X	Store contents of Acc to location X



John von Neumann Machine (3/7)

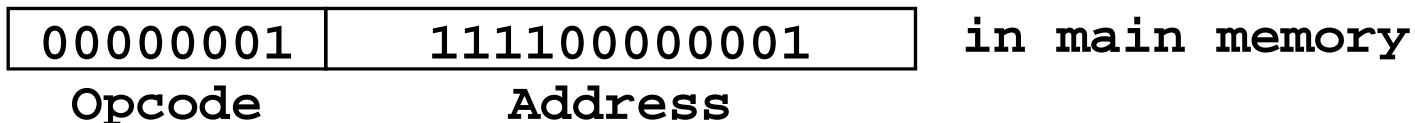
- John von Neumann machine had 20-bit wide instructions



- **Opcode: Operation code.**
Each Opcode corresponds to a basic operation that the CPU performs
- Basic operations such as add, subtract, load, store, *etc.*
- Basic operation → one instruction of machine language
- How many different machine instructions?
 $2^8 = 256$

John von Neumann Machine (4/7)

- How many different memory locations can it address?
- A 12-bit pattern → one address
- $2^{12} = 4096$ or 4K locations (words)
- Load A : Load contents of A to accumulator.
- opcode for load was 00000001
- Suppose variable A is assigned location $(3841)_{10} = (111100000001)_2$
- Thus instruction Load A is stored



- In what location?

John von Neumann Machine (5/7)

- We know how to store program and data in memory
- How do we execute this program?
- By fetching and performing one instruction at a time
- Instruction cycle.

Also known as the '*Fetch – Decode – Execute* cycle'.

- *Fetch* part: Fetch the next instruction as indicated by program counter (PC).
- *Decode* part: The instruction is fetched from memory and stored in IR, the instruction register. Here *opcode* and *address* parts are separated. *opcode* is decoded (in order to understand what needs to be done).

John von Neumann Machine (6/7)

- **Execute Part: It has two parts**
 1. Fetch operand: Depending upon the *opcode*, we fetch the operand using the *address* present in the instruction.
 2. Perform: Actually performing the instruction.
- **Execution of a program means doing *Fetch – Decode – Execute* until we are done.**
- **When do we increment PC?**
- **Right after we fetch. Why?**
- **Do not do it after the *Execute* part.**
- **Jump or Branch instruction**

Jump	< Address >
------	-------------
- **Jump to the address specified.**

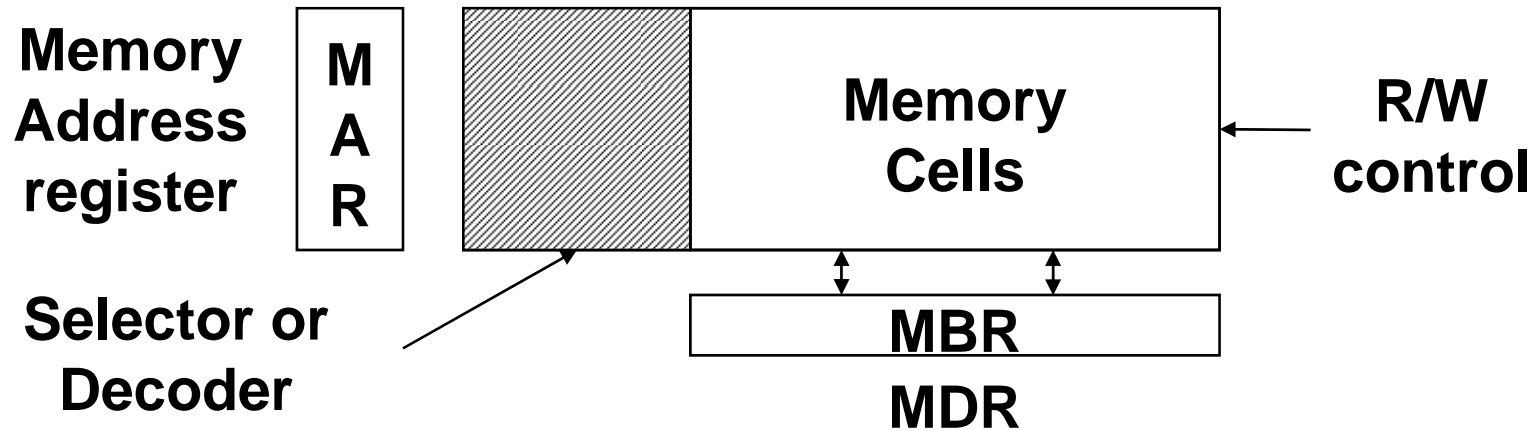
John von Neumann Machine (7/7)

- Execution of Jump instruction:
- Copy the address in the instruction to PC
- Now if *Fetch – Decode – Execute* cycle increments PC at the end, then program will jump to (address+1) and not the (address) specified in the jump instruction.
- We fetch instructions and operands from memory.
- How does CPU (ALU+control) communicate with memory?
- Two registers, MAR & MBR (MDR)
- MAR: Memory Address Register
- MBR: Memory Buffer Register
- MDR: Memory Data Register

Memory Unit (1/3)

- **Memory consists of locations or cells, each of which stores information.**
- **Each cell has an address.**
- **Using its address we refer to that location in our program.**
- **Cell is the smallest addressable unit.**
- **Size of a cell and address of a cell are two different things.**
- **If an address is n -bit long, then 2^n different cells can be addressed.**
- **Most computers (modern) use 8-bit cell (a byte) as the smallest unit.**
- **Suppose my PC has 16MB (megabyte) memory.**
- **How long is $\log_2 16M$? \rightarrow 24 bits for the address.**

Memory Unit (2/3)



- **Size of MAR:** determines how many cells we can have.
Size of MBR (MDR) : memory location size.
- **Idea of a byte was not there when JVN machine was designed.**
- **Read:** send the address to MAR and set control to R. Address is decoded, location is selected, and contents are delivered into MBR. Once data is ready in MBR, it is sent to CPU.
- **Write?** Send address & data; set control to W.

Memory Unit (3/3)

- How do we execute 'Load A' instruction?
- Fetch A: Address from IR (Instruction Register) is sent to MAR, with a read request.
- Perform: Once data is available in MBR/MDR, copy its contents to accumulator (ACC).
- How do we execute 'ADD A'?
- Fetch part same as before.
- Contents of MBR and ACC are sent to ALU for addition.
- Result of this addition is stored in ACC.