CS222: Computer Organization & Assembly Language

Lecture 25:

Programming the Basic Computer

Machine Language

Assembly Language

Pseudo-instructions

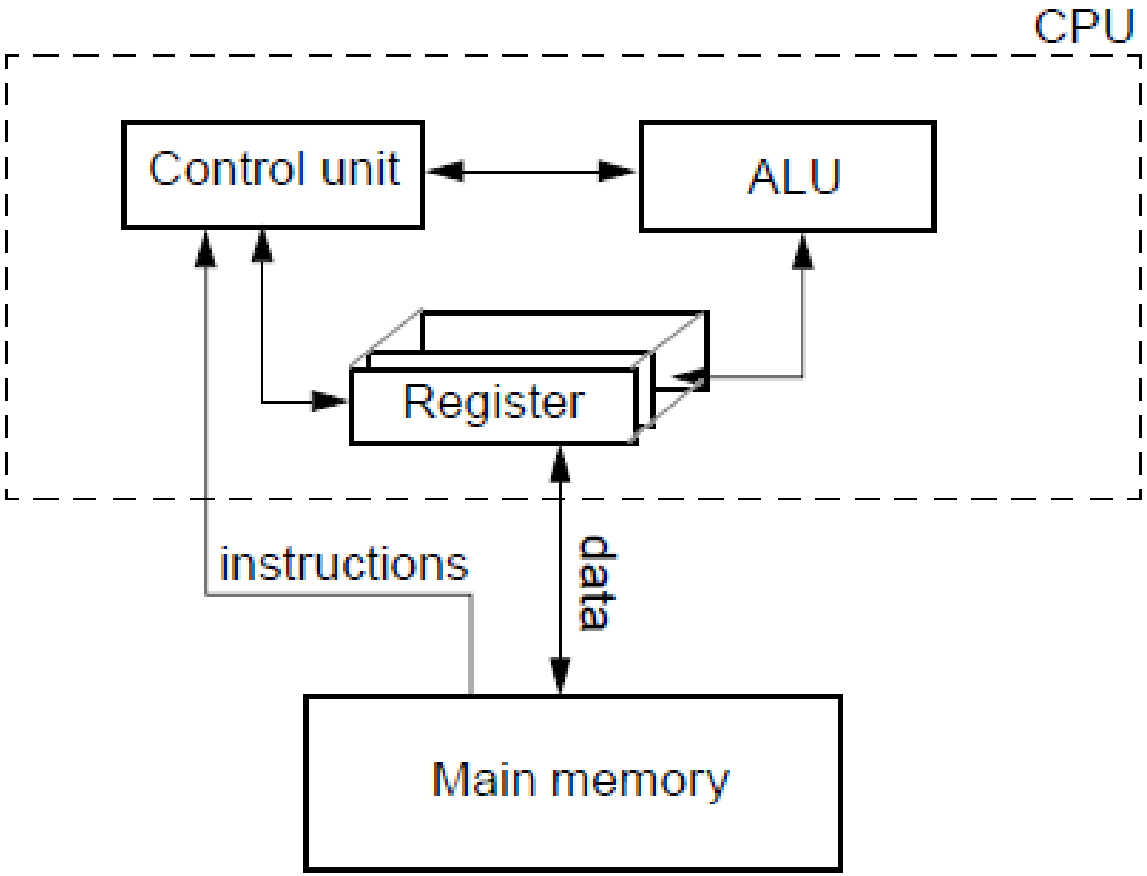
Dr. Ghulam Abbas

Spring-2020

Reading Sections: Computer System Architecture, 3rd Edition by Morris Mano, Sections 6.1 & 6.2

Video Lecture: This lecture is recorded in two parts. Click on a part to view the recording: PART-1, PART-2

THE VON NEUMANN ARCHITECTURE



General-purpose Von Neumann Architecture

DESIGN OF BASIC COMPUTER

Hardware Components of Basic Computer:

A memory unit: 4096 x 16.

Registers:

AR, PC, DR, AC, IR, TR, OUTR, INPR, and SC

Flip-Flops:

I, S, E, R, IEN, FGI, and FGO

Decoders in CU: A 3x8 opcode decoder
A 4x16 timing decoder

ALU: Connected to AC

Common bus: 16 bits

Control logic gates

STEPS OF COMPUTER DESIGN

- **Designing**
  - CPU (ALU, CU)
  - Bus structure
- **Selecting**
  - Memory and Registers
- **Defining**
  - Instruction format
  - Instruction set

BASIC COMPUTER INSTRUCTION SET

Symbol	Hex Code I = 0	Hex Code I = 1	Description
AND	0xxx	8xxx	AND memory word to AC
ADD	1xxx	9xxx	Add memory word to AC
LDA	2xxx	Axxx	Load AC from memory
STA	3xxx	Bxxx	Store content of AC into memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and save return address
ISZ	6xxx	Exxx	Increment and skip if zero
CLA	7800		Clear AC
CLE	7400		Clear E
CMA	7200		Complement AC
CME	7100		Complement E
CIR	7080		Circulate right AC and E
CIL	7040		Circulate left AC and E
INC	7020		Increment AC
SPA	7010		Skip next instr. if AC is positive
SNA	7008		Skip next instr. if AC is negative
SZA	7004		Skip next instr. if AC is zero
SZE	7002		Skip next instr. if E is zero
HLT	7001		Halt computer
INP	F800		Input character to AC
OUT	F400		Output character from AC
SKI	F200		Skip on input flag
SKO	F100		Skip on output flag
ION	F080		Interrupt on
IOF	F040		Interrupt off

PROGRAMMING LANGUAGES

- **Program**

  A collection of instructions and operands for directing the computer to perform a required data processing task

- **Various types of programming languages**

  - **Machine-language**
    - Binary or Hexadecimal code
  - **Assembly-language**
    - Symbolic code (mnemonics)
    - Translation is done by Assembler
  - **High-level language**
    - Translation is done by Complier

COMPARISON OF PROGRAMMING LANGUAGES

- How to add two numbers using C++?

```
int  A, B, C;
A= 3;
B= 5;
C = A + B;
```

- How are the numbers actually added by CPU?
  - Example: 3-bit processor

Instruction Set of 3-bit processor

Stop	0	0	0	STP
Load as Directed	0	0	1	LDD
Load Immediate	0	1	0	LDI
Store as Directed	0	1	1	STD
Add as Directed	1	0	0	ADD
Subtract as	1	0	1	SUB
Directed Jump	1	1	0	JMP
Jump if Zero	1	1	1	JEZ

- Equivalent Program for 3-bit Processor

Mnemonics or symbolic code

Location	Instruction
0	LDD 4
1	ADD 5
2	STD 6
3	STP
4	3 (1st operand)
5	5 (2nd operand)
6	

Machine Language or Binary code

Location	Instruction
0	00100100
1	10000101
2	01100110
3	00000000
4	00000011 (1st operand)
5	00000101 (2nd operand)
6	

BASIC COMPUTER INSTRUCTION SET

Symbol	Hex Code I = 0	Hex Code I = 1	Description
AND	0xxx	8xxx	AND memory word to AC
ADD	1xxx	9xxx	Add memory word to AC
LDA	2xxx	Axxx	Load AC from memory
STA	3xxx	Bxxx	Store content of AC into memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and save return address
ISZ	6xxx	Exxx	Increment and skip if zero
CLA	7800		Clear AC
CLE	7400		Clear E
CMA	7200		Complement AC
CME	7100		Complement E
CIR	7080		Circulate right AC and E
CIL	7040		Circulate left AC and E
INC	7020		Increment AC
SPA	7010		Skip next instr. if AC is positive
SNA	7008		Skip next instr. if AC is negative
SZA	7004		Skip next instr. if AC is zero
SZE	7002		Skip next instr. if E is zero
HLT	7001		Halt computer
INP	F800		Input character to AC
OUT	F400		Output character from AC
SKI	F200		Skip on input flag
SKO	F100		Skip on output flag
ION	F080		Interrupt on
IOF	F040		Interrupt off

COMPARISON OF PROGRAMMING LANGUAGES

- **BC Machine Language Program to Add Two Numbers**

Location	Instruction Code
0	0010 0000 0000 0100
1	0001 0000 0000 0101
10	0011 0000 0000 0110
11	0111 0000 0000 0001
100	0000 0000 0000 0011
101	0000 0000 0000 0101
110	0000 0000 0000 0000

Load from address 100

LDA

- **Hexadecimal program**

Location	Instruction
000	2004
001	1005
002	3006
003	7001
004	3
005	5
006	0000

- **Basic Computer Program using Mnemonics or symbolic code**

Location	Instruction		Comments
000	LDA	004	Load 1st operand into AC
001	ADD	005	Add 2nd operand to AC
002	STA	006	Store sum in location 006
003	HLT		Halt computer
004	3		1st operand
005	5		2nd operand
006	0000		Store sum here

Symbol	*Hex Code I = 0*	*I = 1*	*Description*
ADD	1xxx	9xxx	Add memory word to AC
LDA	2xxx	Axxx	Load AC from memory
STA	3xxx	Bxxx	Store content of AC into memory
HLT	7001		Halt computer

- **Assembly-Language Program (note variable names instead of addresses)**

```
        ORG     0       /Origin of program is location 0
        LDA     A       /Load  operand from location A
        ADD     B       /Add  operand from location B
        STA     C       /Store sum in location C
        HLT             /Halt computer
A,      DEC     3       /Decimal operand
B,      DEC     5       /Decimal operand
C,      DEC     0       /Sum stored in location C
        END             /End of symbolic program
```

ASSEMBLY LANGUAGE

Syntax of the BC assembly language

Each line is arranged in three columns called fields

***Label* field**

- **May be empty or may specify a symbolic address**
- **Consists of up to 3 characters**
- **Terminated by a comma**

***Instruction* field**

- **Specifies a machine instruction or a pseudo instruction**
- **May specify one of**
  - *** Memory reference instr. (MRI)**
    MRI consists of two or three symbols separated by spaces.
    ADD A (direct address MRI)
    ADD A I (indirect address MRI)
  - *** Register reference or input-output instr.**
    Non-MRI does not have an address part
  - *** Pseudo instr. with or without an operand**

***Comment* field**

- **May be empty or may include a comment**

PSEUDO-INSTRUCTIONS

ORG N: Hexadecimal number N is the memory location for the instruction or operand listed in the following line

END: Denotes the end of symbolic program

DEC N: Signed decimal number N to be converted to the binary

HEX N: Hexadecimal number N to be converted to the binary

SIMULATING THE BASIC COMPUTER

- Students must use the Basic Computer Simulator to practice Assembly Language Programming.
  - The Basic Computer Simulator is the courtesy of MIT Licence.

- Please use the ***simulator_basic_comp.swf*** file uploaded to the Drobox to access the simulator.
  - The uploaded “flash projector” may be used to open the *.swf* file.
    - » Press Ctrl+F to view all registers, flip-flops and memory.
  - Alternatively, use your web browser to open the .swf file.

DIFFERENCE IN SYNTAX

- Note the following differences in the syntax of the actual assembly language (given in the book) and the one used in the simulator:
  - A maximum of three characters are allowed in the label field in the book, whereas, in the simulator, the number of characters can be large.
  - The label field is terminated with a "**,**" in the book, whereas in the simulator "**:**" is used.
  - The comment field in the book starts with "/", whereas in the simulator it starts with "//".