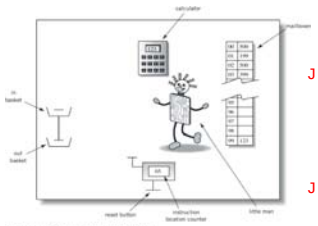
The Little Man Computer

The Little Man Computer: Components & Physical Layout



Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition Chapter 6, Figure 06-01

- 100 mailboxes
  - address 00 - 99
  - Each holds 3-digit decimal number
- Calculator
  - Enter number
  - Temporarily hold number
  - Add and Subtraction
  - Display 3 digits
- Hand counter
  - Reset button
  - Instruction counter
- Little Man
- In/Out Basket

The Little Man Computer: Components & Physical Layout

- **Start Button:**

The operator of the LMC (not the Little Man) presses the START BUTTON to zero the Instruction Counter and ring a bell which wakes up the Little Man so he can start executing the program which has previously been stored in the mailboxes

- **Instruction Counter:**

This display contains the number of the mailbox in which the NEXT instruction is stored. The Little Man can change the number in the instruction counter (usually by incrementing but sometimes by replacement), and the user of the LMC can reset the Instruction Counter to 00 using the start button

The Little Man Computer: Components & Physical Layout

- **Calculator:**

Performs simple arithmetic (addition and subtraction only)

Used for temporary storage of a single three digit number. The display is limited to three digits.

The calculator has ten numerical keys (0-9) and two operation keys (+ and -).

The calculator also has two lights which can be seen by the Little Man.

One of these lights turns on whenever the number being displayed is exactly zero.

The other light turns on whenever the number being displayed is positive (the number ZERO is considered a positive number). Thus when ZERO is displayed on the calculator, both lights will be on.

These lights are sometimes referred to as flags

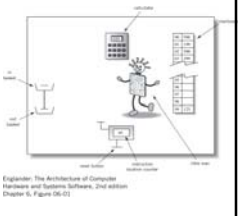
The Little Man Computer: Components & Physical Layout

- **Mailboxes:**

Mailboxes are identified using two-digit numbers (00 99) and each can hold a single slip of paper containing a single three digit number.

The Little Man Computer: Communication with outside World

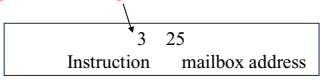
- A user outside the mailroom communicate with the little man by
  - Putting a 3-digit into the in basket
  - Retrieving a 3-digit from the out basket
- Apart from the reset button, all communication between the LMC and the outside world takes place using 3-digit numbers



The Little Man Computer: Operation

- A small group of instructions, each consists of a single digit
- The first digit of a 3-digit number is used to tell the Little Man which operation to perform.
- In some cases, the Little Man is required to use a particular mailbox to store or retrieve data
- The 3 digits can be used to give instructions to the Little Man according to

Operation Code (Op-Code)



The Little Man Computer: Instruction Set

FORMAT	MNEMONIC	MEANING
3xx	STO xx	Stores the calculator value into mailbox xx.
4xx	STA xx	Stores the address portion of the calculator value (last 2 digits) into the address portion of the instruction in mailbox xx.
5xx	LOAD xx	Loads the contents of mailbox xx into the calculator.

The Little Man Computer: Instruction Set

FORMAT	MNEMONIC	MEANING
1xx	ADD xx	Adds the contents of mailbox xx to the calculator display.
2xx	SUB xx	Subtracts the contents of mailbox xx from the calculator display.

The Little Man Computer: Instruction Set

FORMAT	MNEMONIC	MEANING
000	STOP	Stops the Computer - the Little Man rests.
6xx	B xx	This instruction sets the instruction counter to the number xx, thus effectively branching to mailbox xx
7xx	BZ xx	IF the calculator value is zero, THEN set the instruction counter to the number xx, thus effectively branching to mailbox xx.
8xx	BP xx	IF the calculator value is positive, THEN set the instruction counter to the number xx, thus effectively branching to mailbox xx. NOTE: zero is considered positive.

The Little Man Computer: Instruction Set

FORMAT	MNEMONIC	MEANING
901	INPUT	Read a number from the IN basket and key it into the calculator.
902	OUTPUT	Copy the number in the calculator onto a slip of paper and place it into the OUT basket.

The Little Man Computer: Execute Program

To actually run an LMC program we must carry out the following steps.

- Load the instructions into the mailboxes, starting with mailbox 00.
- Place the data to be used by the program in the IN basket, in the order in which the program will use these data.
- Press the RESET BUTTON to set the Instruction Counter to 00 and to also wakeup the Little Man. Then

1. The Little Man looks into the Instruction Counter
2. The Little Man find the mailbox whose address has the value in the Instruction Counter and fetches the 3-digit number from the mailbox.
3. The Little Man executes the instruction found in the mailbox
4. The Little Man go over to increase the instruction counter by 1
5. The Little Man repeat the by going to 1

The Little Man Computer: Program Example

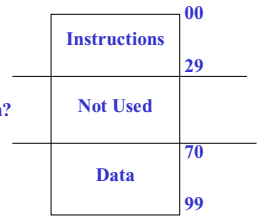
Mail Box Division:

Both Program and data are

stored in the mail box

Which is data, which is program?

Mail Box Map
(Memory Map)



The Little Man Computer: Program Example

Write a LMC program which adds two numbers together

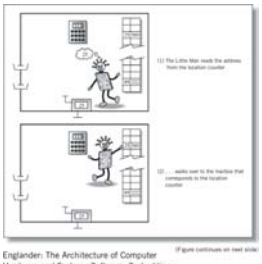
Mailbox	Code	Instruction Description
00	**901**	**INPUT**
01	**399**	**STORE DATA (to mailbox no 99)**
02	**901**	**INPUT**
03	**199**	**ADD the 1st number to 2nd number**
04	**902**	**OUTPUT RESULT**
05	**000**	**The Little Man rest**
99		**Data**

The Little Man Computer: Program Example

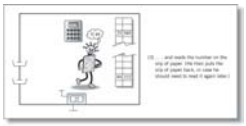
Write a LMC program to find the positive (absolute) difference of two numbers

Mailbox	Mnemonic	Code
00	**IN**	**901**
01	**STO 10**	**310**
02	**IN**	**901**
03	**STO 11**	**311**
04	**SUB 10**	**210**
05	**BP 08**	**808**
06	**LOAD 10**	**510**
07	**SUB 11**	**211**
08	**OUT**	**902**
09	**HALT**	**000**
10	**DAT 00**	**000**
11	**DAT00**	**000**

The Little Man Computer: The Instruction Cycle



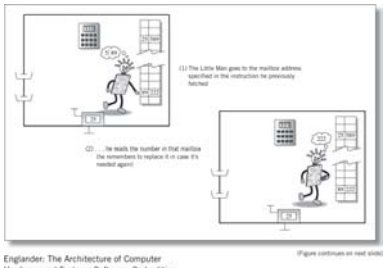
Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition Chapter 6, Figure 06-05a



Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition Chapter 6, Figure 06-05a continued

The fetch portion of the instruction cycle (LOAD Instruction)

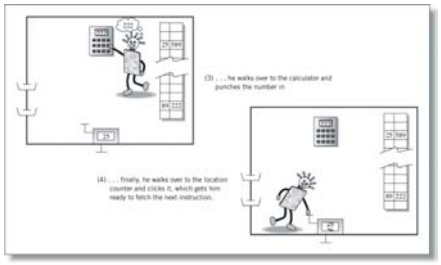
The Little Man Computer: The Instruction Cycle



Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition Chapter 6, Figure 06-05b

The execute portion of the instruction cycle (LOAD instruction)

The Little Man Computer: The Instruction Cycle



Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition Chapter 6, Figure 06-05b continued

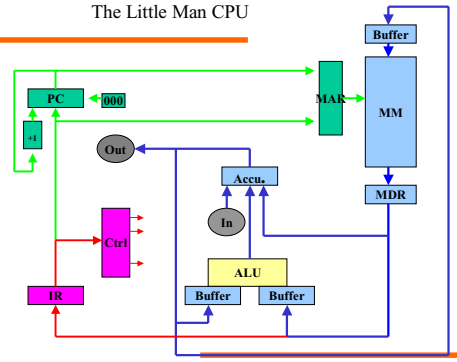
The execute portion of the instruction cycle (LOAD instruction)

Some Observations Regarding Computer Architecture

- Between 1945 and 1951, von Neumann architecture
- Other experimental architectures have been developed and built
- von Neumann architecture continues to be the standard architecture for computer
- No other architecture has had any commercial success so far
- It is significant that in a field where technological changes happens almost overnight, the architecture of computers is virtually unchanged since 1951

- Key concepts of von Neumann architecture include
  - Stored program concept
  - Memory is addressed linearly
  - Memory addressed by the location number without regard to the contents
  - Instructions executed in sequence unless an instruction or outside events cause branch

The Little Man CPU



Assembly Languages And Programming Paradigm

Programming Languages

Divided into two broad categories

– Low-level (close to hardware)

– High-level (abstracted away from hardware)

Characteristics Of High-Level Language

Many-to-one translation

Hardware independence

Application orientation

General-purpose

Powerful abstractions

Characteristics Of Low-Level Language

One-to-one translation

Hardware dependence

Systems programming orientation

Special-purpose

Few abstractions

Terminology

Assembly language

– Refers to a type of low-level language

– Specific to given processor

Assembler

– Refers to software that translates assembly language into binary code

– Analogous to compiler

An Important Concept

Because an assembly language is a low-level language that incorporates specific characteristics of a processor, such as the instruction set, operand addressing, and registers, many assembly languages exist.

Assembly Languages

Share same general structure

Programmer who understands one assembly language can learn another quickly

Assembly Statement Format

General format is:

label: opcode operand1, operand2, ...

Label is optional

Opcode and operands are processor specific

Comment Syntax

Typically

– Character reserved to start a comment

– Comment extends to end of line

Examples of comment characters

– Pound sign (#)

– Semicolon (;)

An LMC Assembly Program

LMC Assembly Program to find the positive (absolute) difference of two numbers

```
        IN      #input 1st number
        STO dat1    #store first number in location dat11
        IN      #input 2nd number
        STO dat2    #store in location dat2
        SUB dat1    #A = A – (dat1)
        BP Label1   #If A>=0, jump to Label1
        LOAD dat1   #load content from location dat1
        SUB dat2    #A – A – (dat2)
Label1  OUT     #output content in A
        HALT    #Halt the program
dat1    DAT 00      #Location reserved for 1st number
dat2    DAT00       #Location reserve for 2nd number
```

Assembly Language For Conditional Execution

if (condition) {

body

}

next statement

code to test condition and
set condition code
branch not true to label
code to perform body
label: code for next statement

In Practice

Because writing application programs in assembly language is difficult, assembly language is reserved for situations where a high-level language has insufficient functionality or results in poor performance.

Assembler

- Software component
- Accepts assembly language program as input
- Produces binary form of program as output
- Uses *two-pass* algorithm

Difference Between Assembler And Compiler

Although both a compiler and an assembler translate a source program into equivalent binary code, a compiler has more freedom to choose which values are kept in registers, the instructions used to implement each statement, and the allocation of variables to memory. An assembler merely provides a one-to-one translation of each statement in the source program to the equivalent binary form.

What An Assembler Provides

- Statements are 1-1 with instructions
- Assembler
  - – Computes relative location for each label
  - – Fills in branch offsets automatically
- Consequence: can insert or delete statements without recomputing offsets manually

Summary

- Assembly language is low-level and incorporates details of a specific processor
- Many assembly languages exist, one per processor
- Each assembly language statement corresponds to one machine instruction
- Same basic programming paradigm used in most assembly languages
- Programmers must code assembly language equivalents of abstractions such as
  - – Conditional execution
  - – Definite and indefinite iteration
  - – Procedure call

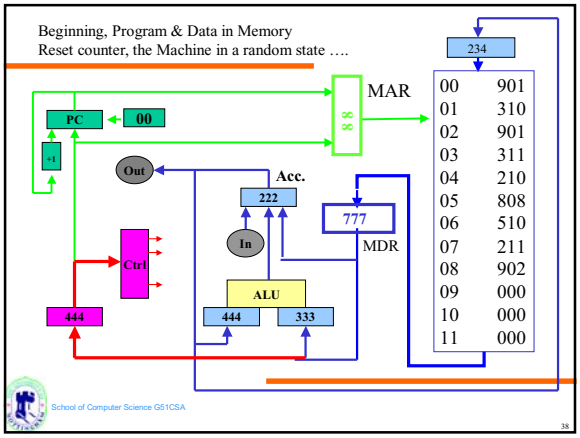
Summary

- Assembler translates assembly language program into binary code
- Assembler uses two-pass processing
  - – First pass assigns relative locations
  - – Second pass generates code

School of Computer Science G51CSA

37

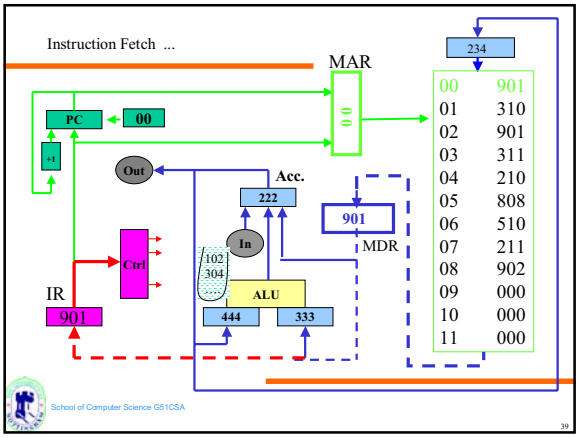
Beginning, Program & Data in Memory
Reset counter, the Machine in a random state ….



School of Computer Science G51CSA

38

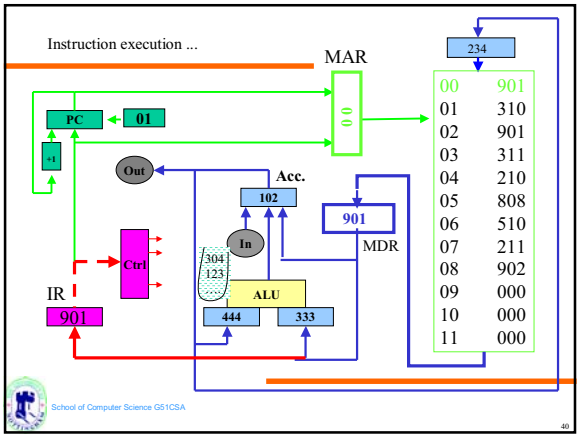
Instruction Fetch …



School of Computer Science G51CSA

39

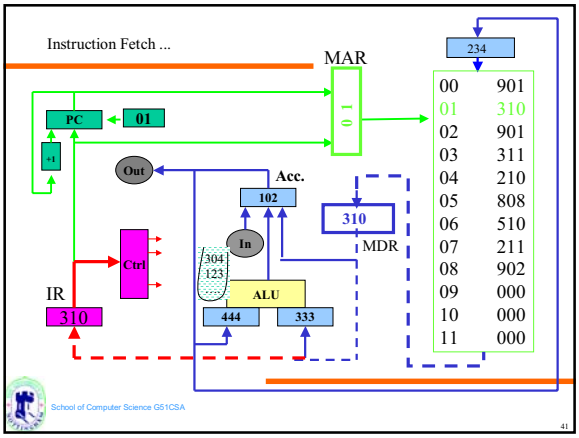
Instruction execution …



School of Computer Science G51CSA

40

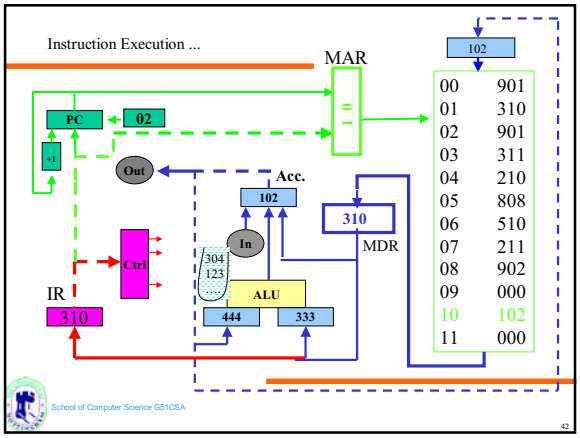
Instruction Fetch …



School of Computer Science G51CSA

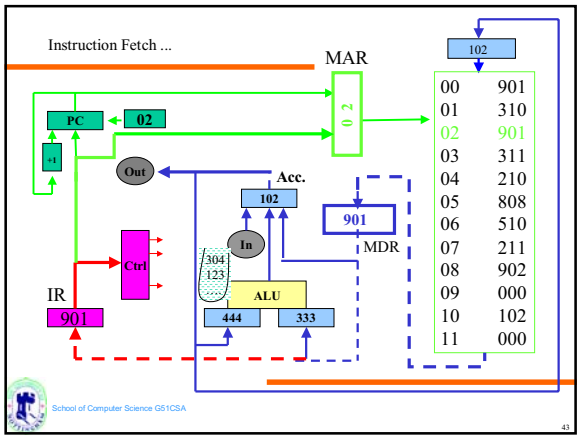
41

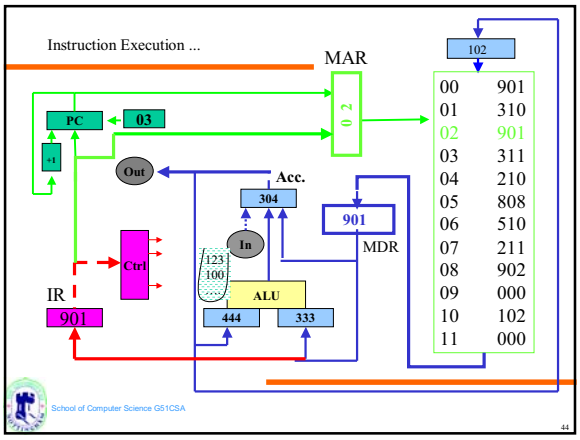
Instruction Execution …

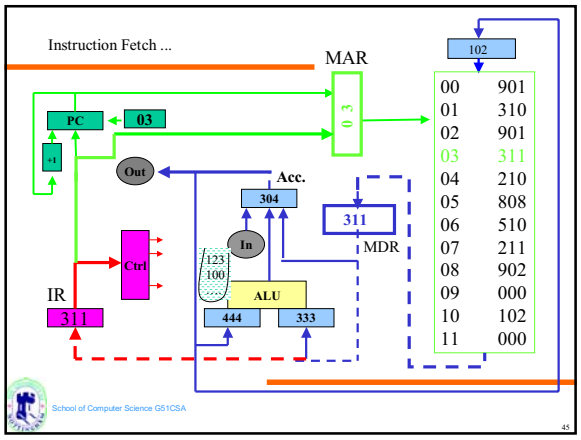


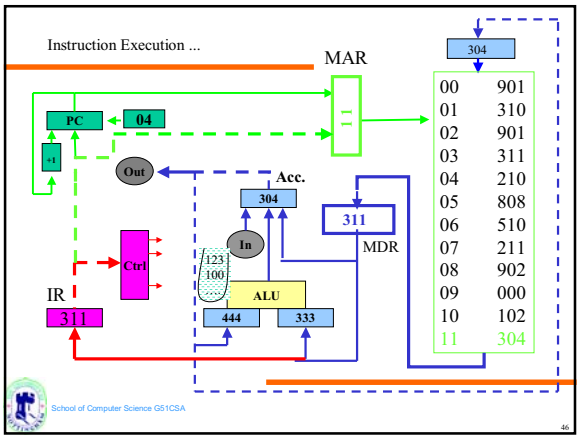
School of Computer Science G51CSA

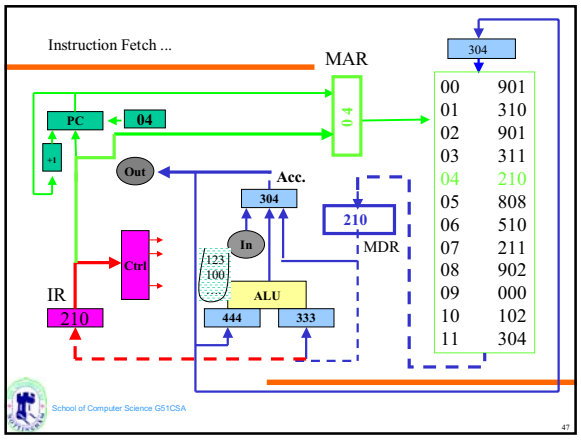
42

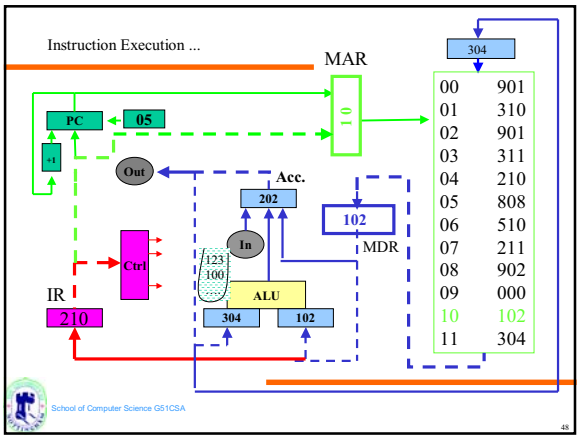

Instruction Fetch ...
MAR
PC
02
+1
Out
Acc.
102
In
304
123
ALU
444
333
901
MDR
Ctrl
IR
901
102
00 901
01 310
02 901
03 311
04 210
05 808
06 510
07 211
08 902
09 000
10 102
11 000
School of Computer Science G51CSA
43

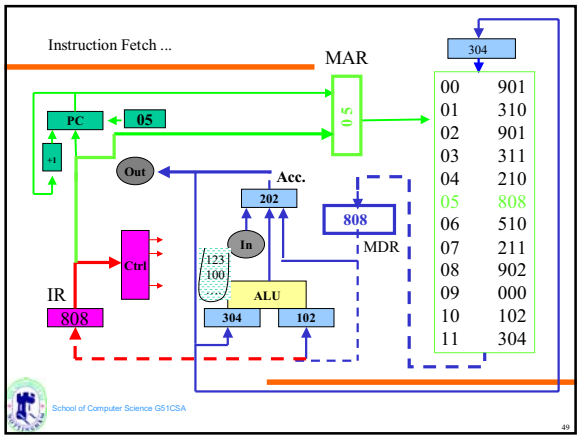

Instruction Execution ...
MAR
PC
03
+1
Out
Acc.
304
In
123
100
ALU
444
333
901
MDR
Ctrl
IR
901
102
00 901
01 310
02 901
03 311
04 210
05 808
06 510
07 211
08 902
09 000
10 102
11 000
School of Computer Science G51CSA
44

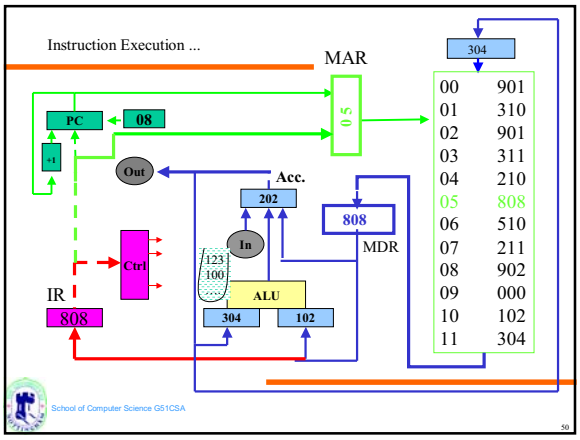

Instruction Fetch ...
MAR
PC
03
+1
Out
Acc.
304
In
123
100
ALU
444
333
311
MDR
Ctrl
IR
311
102
00 901
01 310
02 901
03 311
04 210
05 808
06 510
07 211
08 902
09 000
10 102
11 000
School of Computer Science G51CSA
45

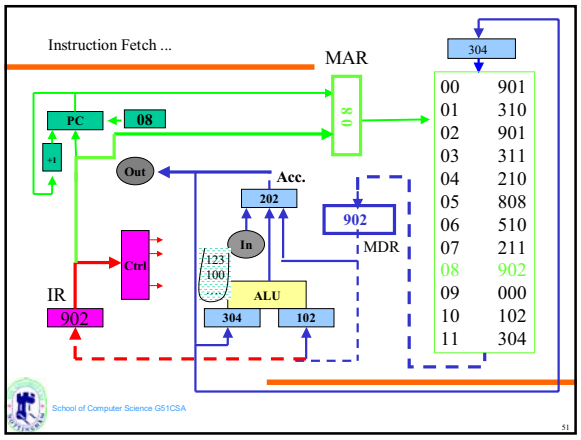

Instruction Execution ...
MAR
PC
04
+1
Out
Acc.
304
In
123
100
ALU
444
333
311
MDR
Ctrl
IR
311
304
00 901
01 310
02 901
03 311
04 210
05 808
06 510
07 211
08 902
09 000
10 102
11 304
School of Computer Science G51CSA
46

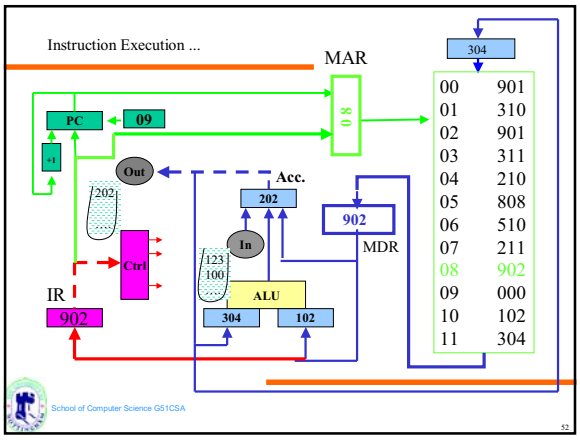

Instruction Fetch ...
MAR
PC
04
+1
Out
Acc.
304
In
123
100
ALU
444
333
210
MDR
Ctrl
IR
210
304
00 901
01 310
02 901
03 311
04 210
05 808
06 510
07 211
08 902
09 000
10 102
11 304
School of Computer Science G51CSA
47

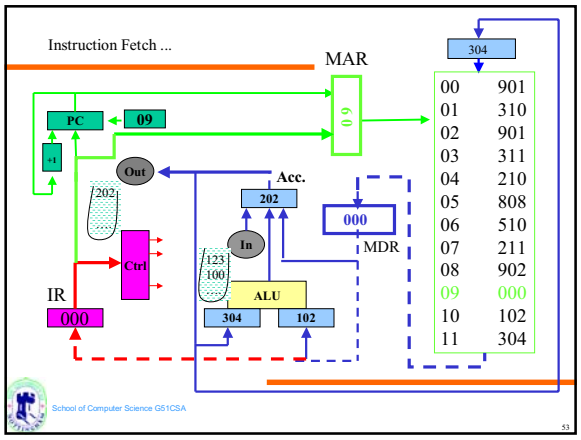

Instruction Execution ...
MAR
PC
05
+1
Out
Acc.
202
In
123
100
ALU
304
102
102
MDR
Ctrl
IR
210
304
00 901
01 310
02 901
03 311
04 210
05 808
06 510
07 211
08 902
09 000
10 102
11 304
School of Computer Science G51CSA
48

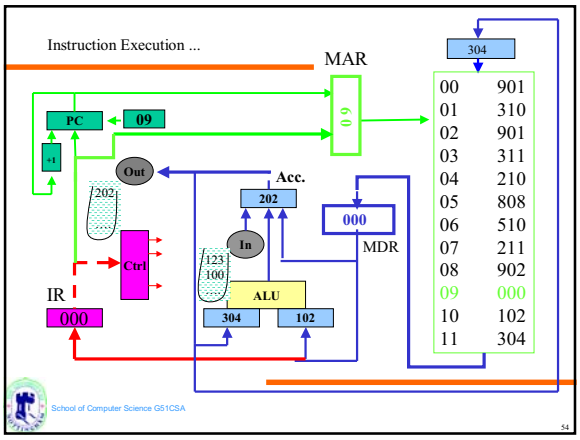

Instruction Fetch ...
MAR
304
PC
05
+1
Out
Acc.
202
808
MDR
In
Ctrl
123
100
ALU
IR
808
304
102
00 901
01 310
02 901
03 311
04 210
05 808
06 510
07 211
08 902
09 000
10 102
11 304
School of Computer Science G51CSA
49


Instruction Execution ...
MAR
304
PC
08
+1
Out
Acc.
202
808
MDR
In
Ctrl
123
100
ALU
IR
808
304
102
00 901
01 310
02 901
03 311
04 210
05 808
06 510
07 211
08 902
09 000
10 102
11 304
School of Computer Science G51CSA
50


Instruction Fetch ...
MAR
304
PC
08
+1
Out
Acc.
202
902
MDR
In
Ctrl
123
100
ALU
IR
902
304
102
00 901
01 310
02 901
03 311
04 210
05 808
06 510
07 211
08 902
09 000
10 102
11 304
School of Computer Science G51CSA
51


Instruction Execution ...
MAR
304
PC
09
+1
Out
Acc.
202
902
MDR
In
Ctrl
123
100
ALU
IR
902
304
102
00 901
01 310
02 901
03 311
04 210
05 808
06 510
07 211
08 902
09 000
10 102
11 304
School of Computer Science G51CSA
52


Instruction Fetch ...
MAR
304
PC
09
+1
Out
Acc.
202
000
MDR
In
Ctrl
123
100
ALU
IR
000
304
102
00 901
01 310
02 901
03 311
04 210
05 808
06 510
07 211
08 902
09 000
10 102
11 304
School of Computer Science G51CSA
53


Instruction Execution ...
MAR
304
PC
09
+1
Out
Acc.
202
000
MDR
In
Ctrl
123
100
ALU
IR
000
304
102
00 901
01 310
02 901
03 311
04 210
05 808
06 510
07 211
08 902
09 000
10 102
11 304
School of Computer Science G51CSA
54

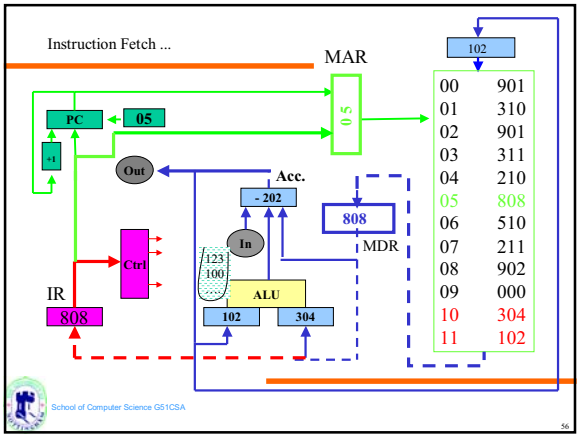
If the first number is bigger than the first number

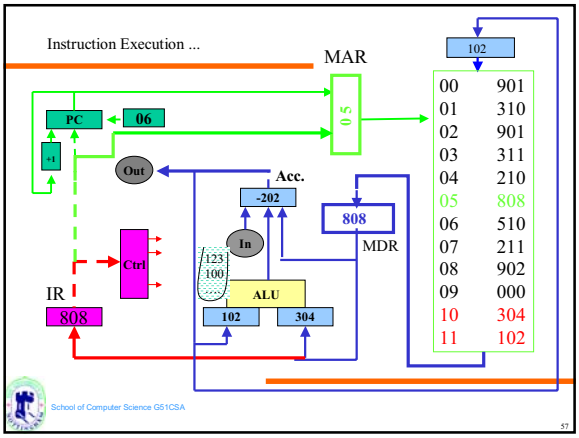
……., e.g, the input basket looks like this

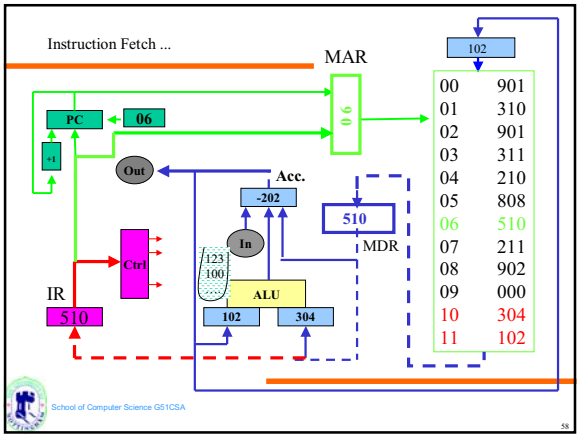
not this

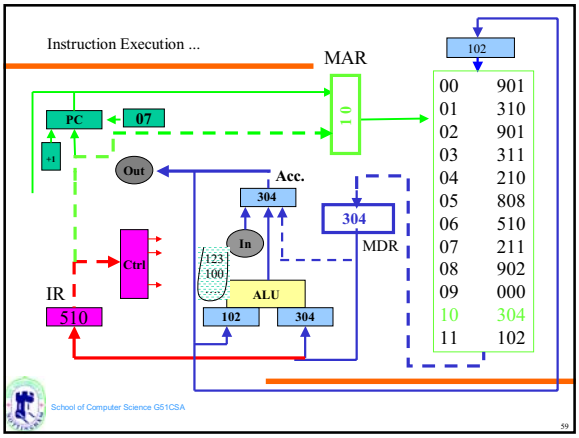
School of Computer Science G51CSA

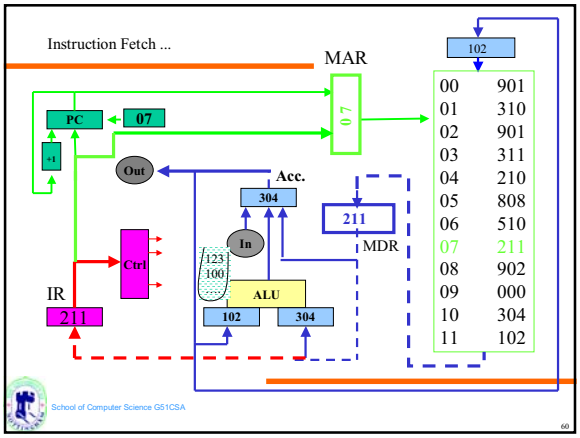
55

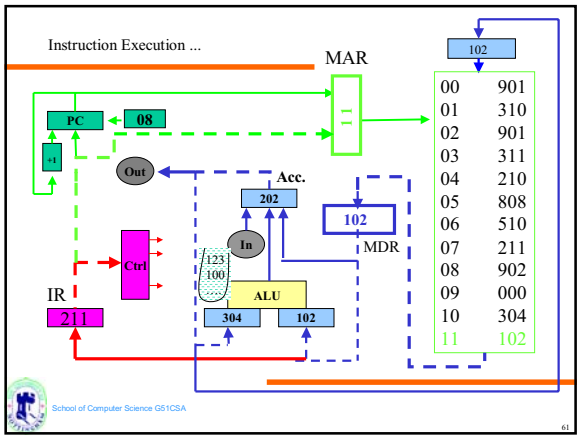


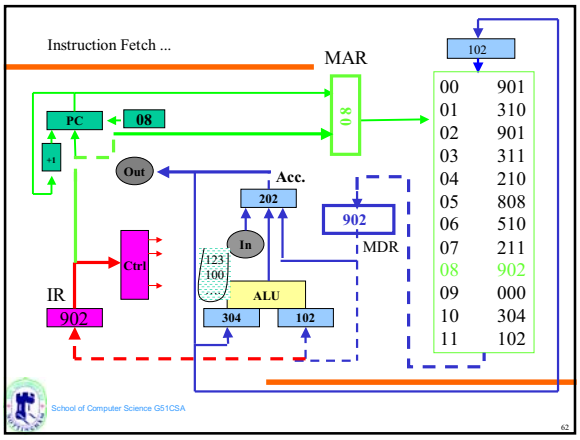


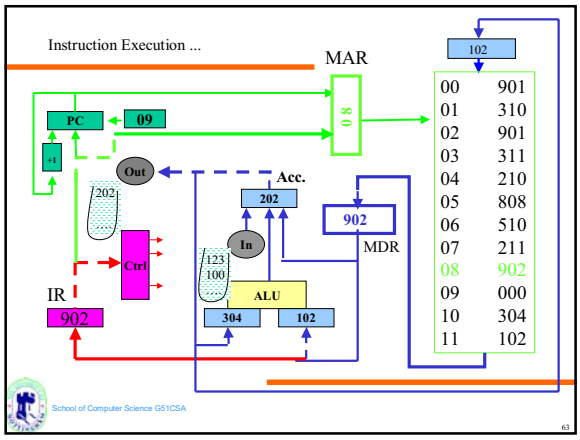


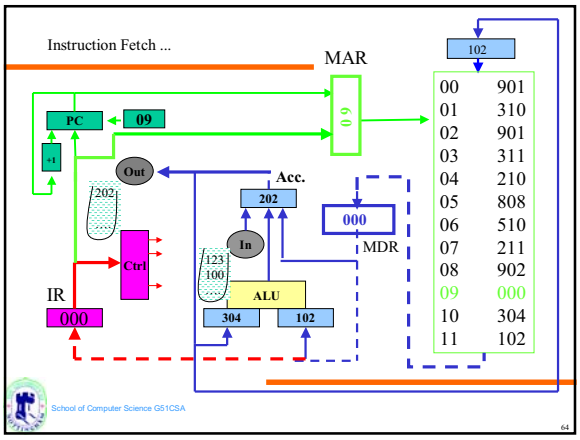


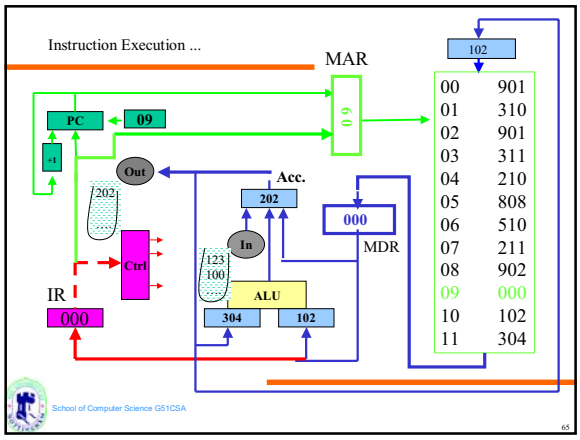



Instruction Execution ...
MAR
11
PC
08
+1
Out
Acc.
202
102
MDR
In
123
100
Ctrl
ALU
IR
211
304
102
102
00 901
01 310
02 901
03 311
04 210
05 808
06 510
07 211
08 902
09 000
10 304
11 102
School of Computer Science G51CSA
61


Instruction Fetch ...
MAR
08
PC
08
+1
Out
Acc.
202
902
MDR
In
123
100
Ctrl
ALU
IR
902
304
102
102
00 901
01 310
02 901
03 311
04 210
05 808
06 510
07 211
08 902
09 000
10 304
11 102
School of Computer Science G51CSA
62


Instruction Execution ...
MAR
09
PC
09
+1
Out
Acc.
202
902
MDR
In
123
100
Ctrl
ALU
IR
902
304
102
102
00 901
01 310
02 901
03 311
04 210
05 808
06 510
07 211
08 902
09 000
10 304
11 102
School of Computer Science G51CSA
63


Instruction Fetch ...
MAR
09
PC
09
+1
Out
Acc.
202
000
MDR
In
123
100
Ctrl
ALU
IR
000
304
102
102
00 901
01 310
02 901
03 311
04 210
05 808
06 510
07 211
08 902
09 000
10 304
11 102
School of Computer Science G51CSA
64


Instruction Execution ...
MAR
09
PC
09
+1
Out
Acc.
202
000
MDR
In
123
100
Ctrl
ALU
IR
000
304
102
102
00 901
01 310
02 901
03 311
04 210
05 808
06 510
07 211
08 902
09 000
10 102
11 304
School of Computer Science G51CSA
65