

Review of the basic of Computer architecture and Operating system

Lecture 2

von Neumann Architecture (1945)

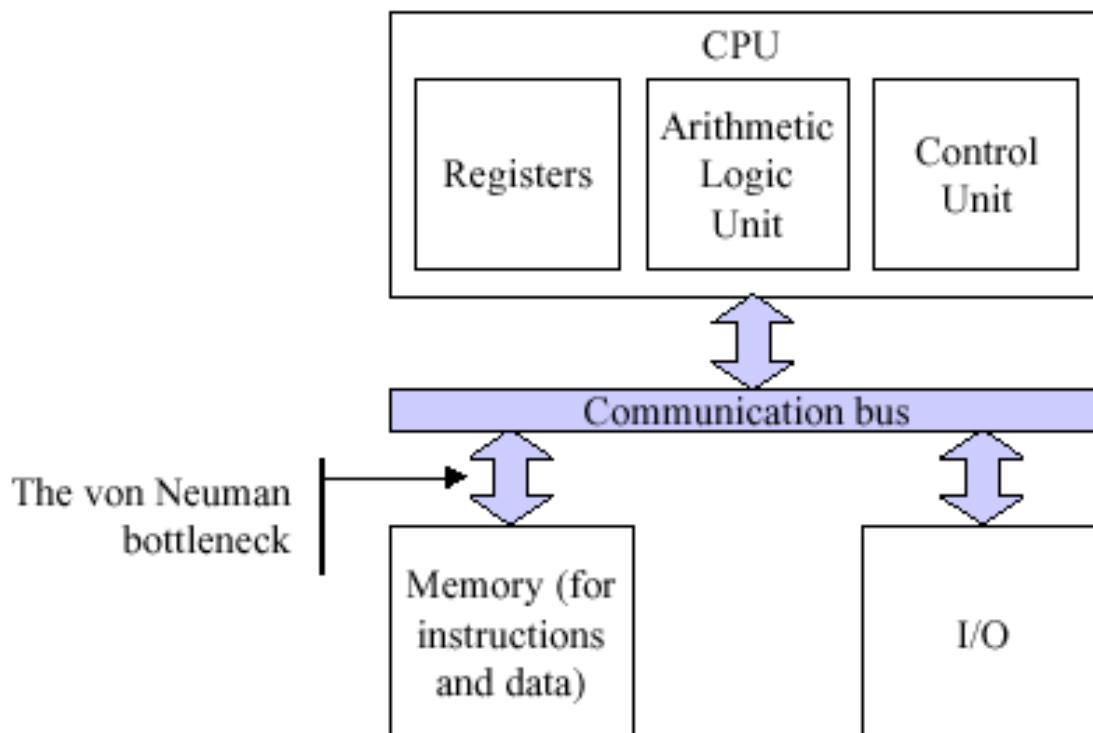
- John Von Neumann is usually considered to be the developer of modern computer architecture.
- The major guidelines that define a Von Neumann architecture are:
 - Stored program concept - memory holds both programs and data.
 - Memory is addressed linearly – i.e address consists of a single number
 - Memory is addressed without regard to content i.e. it can be an instruction or data.
 - Instructions are generally executed sequentially....

Von Neumann Architecture

- Von Neumann defined the functional organisation of the computer to be made up of:
 1. A *control unit* that executes instructions
 2. An *arithmetic logic unit* that performs arithmetic and logical calculations,
 3. Memory locations
 4. Input/output
 5. Communication bus

Structure of a Simple Computer

The abstract *von Neumann architecture* is used in most computers...



An alternative is the *Harvard architecture* that uses separate buses and memory for program instructions and data...

Contents of Memory locations

- Op code
 - Operation code
 - Arbitrary mnemonic
- Operand
 - Object to be manipulated
 - Data or
 - Address of data

Address	Content	
	Op code	Operand

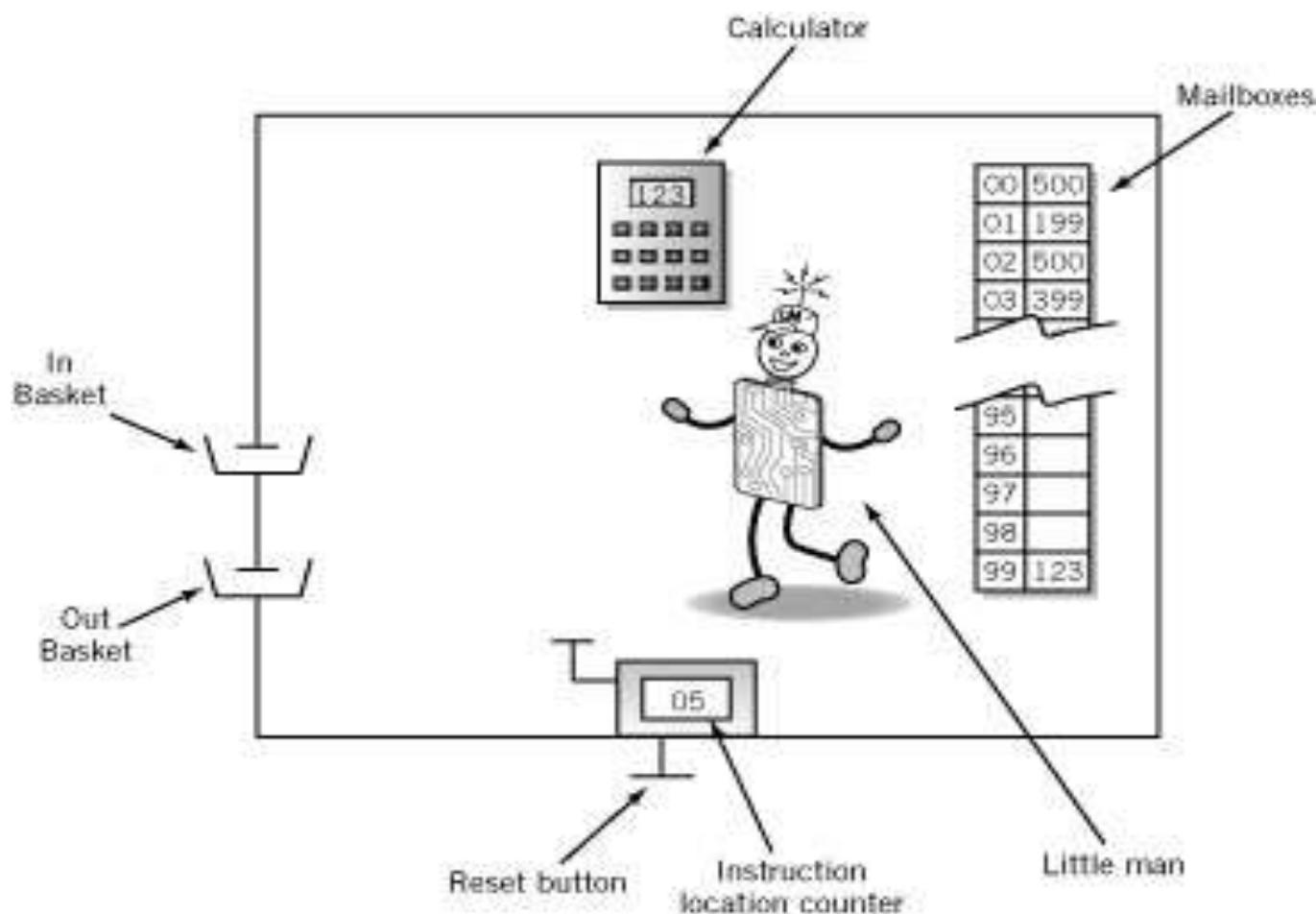
Sample Machine code Instruction Set

Arithmetic	1xx	ADD
	2xx	SUBTRACT
Data Movement	3xx	STORE
	5xx	LOAD
Input/Output	901	INPUT
	902	OUTPUT
Machine Control (coffee break)	000	STOP

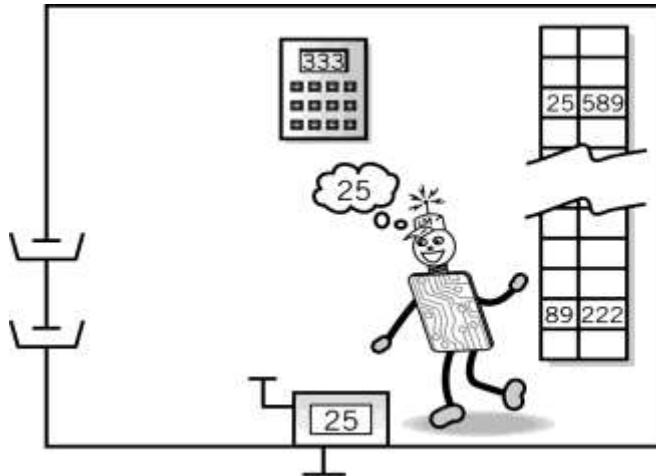
Instruction Cycle

- *Fetch*: Little Man finds out what instruction he is to execute
- *Execute*: Little Man performs the work he is instructed to perform
- The following shows the operation of the **load instruction**

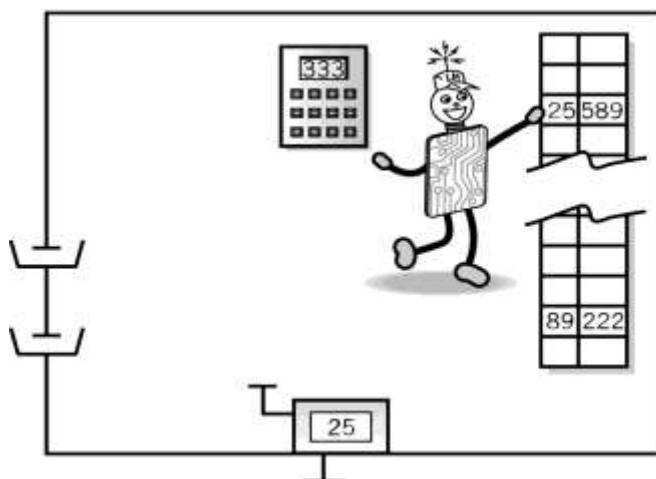
The Little Man Computer



Fetch Portion of Fetch and Execute Cycle

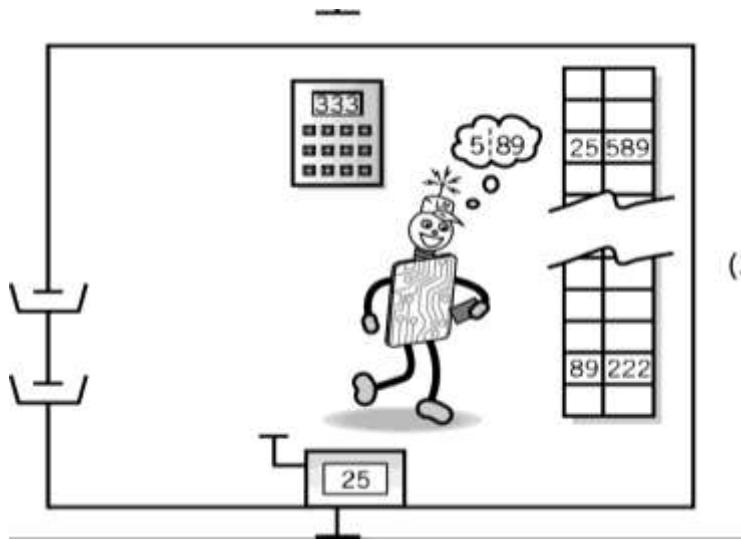


1. Little Man reads the address from the location counter



2. He walks over to the mailbox that corresponds to the location counter

Fetch, cont.

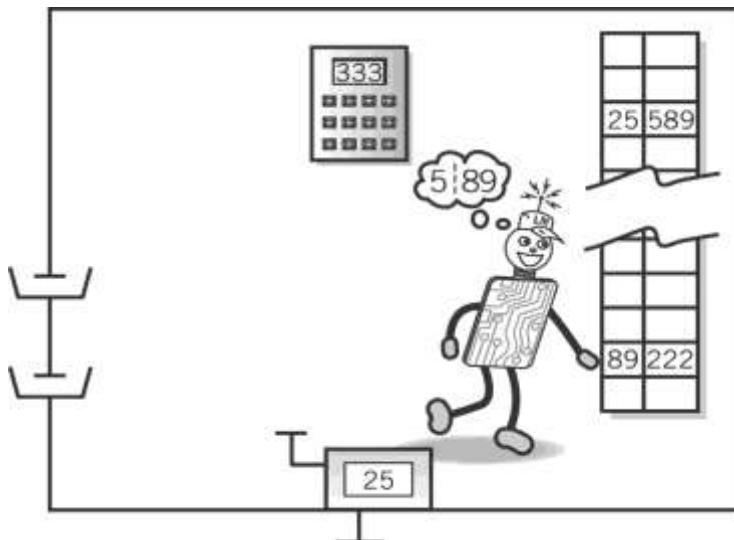


3. And reads (involves knowing the meaning of the op code...) the contents on the slip of paper (he puts the slip back in case he needs to read it again later)

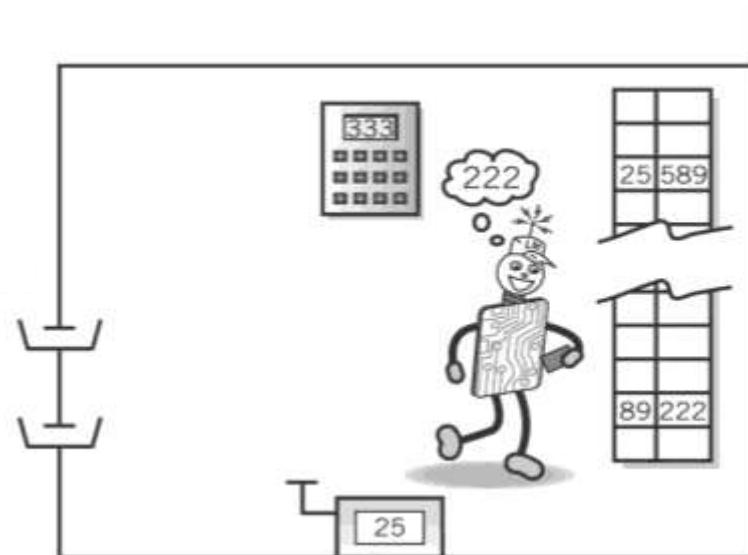
Execute Portion

- The execution portion of each instruction is, of course, different for each instruction.
- However, even here, there are many similarities.
- The load instruction (LDA) is typical. (589: where 5 is the load op-code and 89 refers to the mail box whose contents are loaded into calculator)

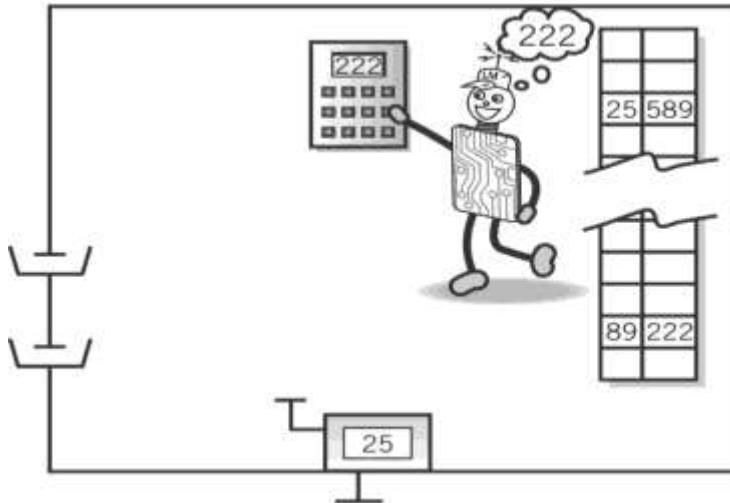
Execute Portion (LDA)



1. The Little Man goes to the mailbox address specified in the instruction he just fetched.
2. He reads the number in that mailbox (he remembers to replace it in case he needs it later).

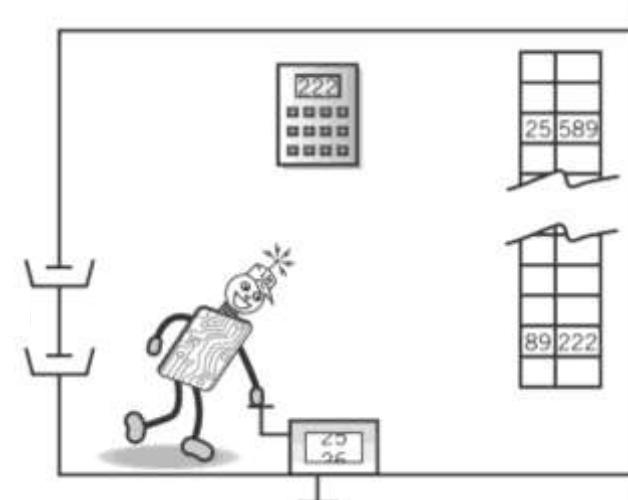


Execute, cont.

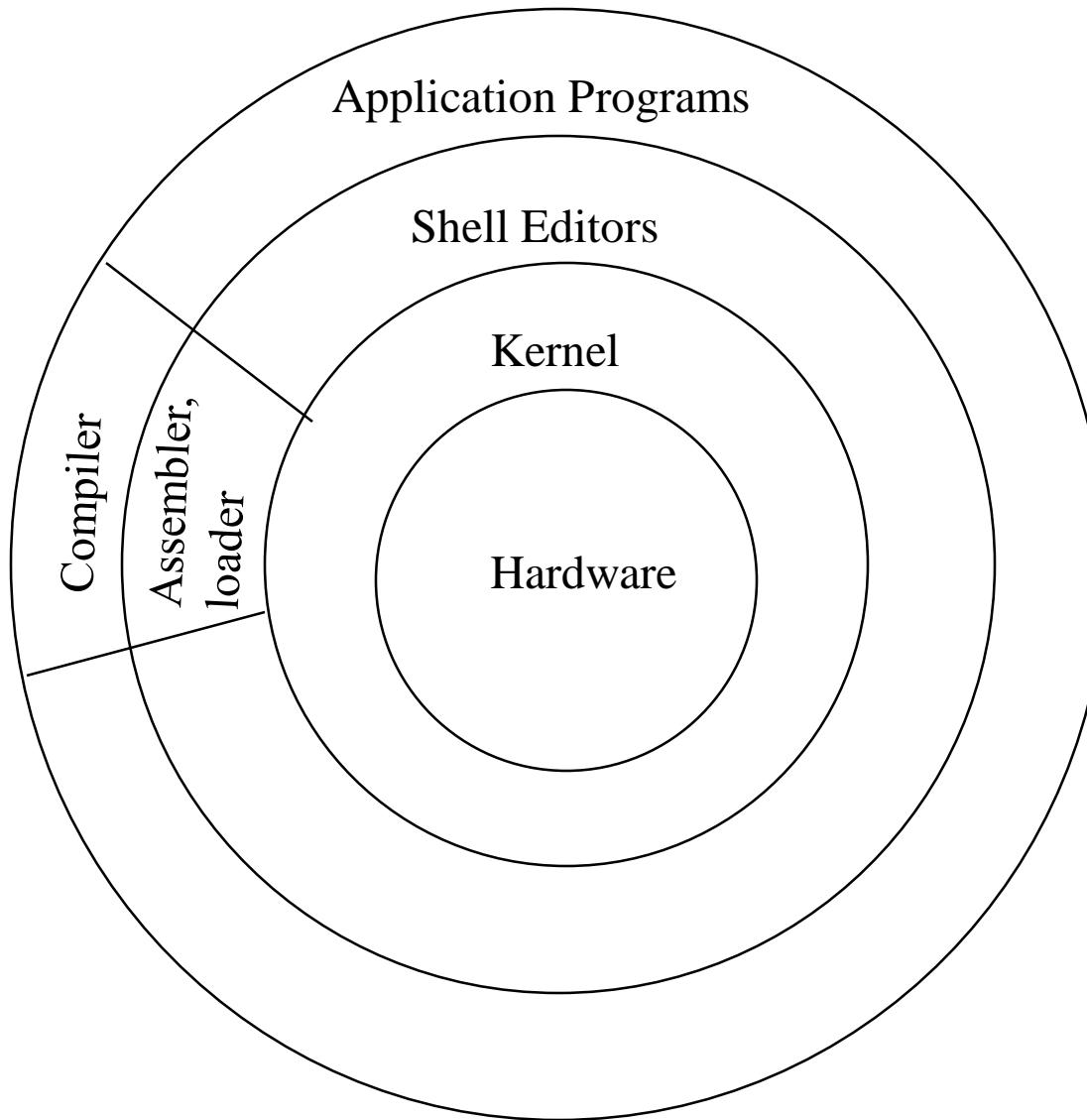


3. He walks over to the calculator and punches the number in.

4. He walks over to the location counter and clicks it, which gets him ready to fetch the next instruction.



An O.S. Architecture



An O.S. Architecture

- The hardware is *encapsulated* by the kernel
- It provides systems services to application programs
- The user communicates with the kernel via the shell
- The user can invoke the C compiler to create applications which can be executed by the kernel

What is an Operating System?

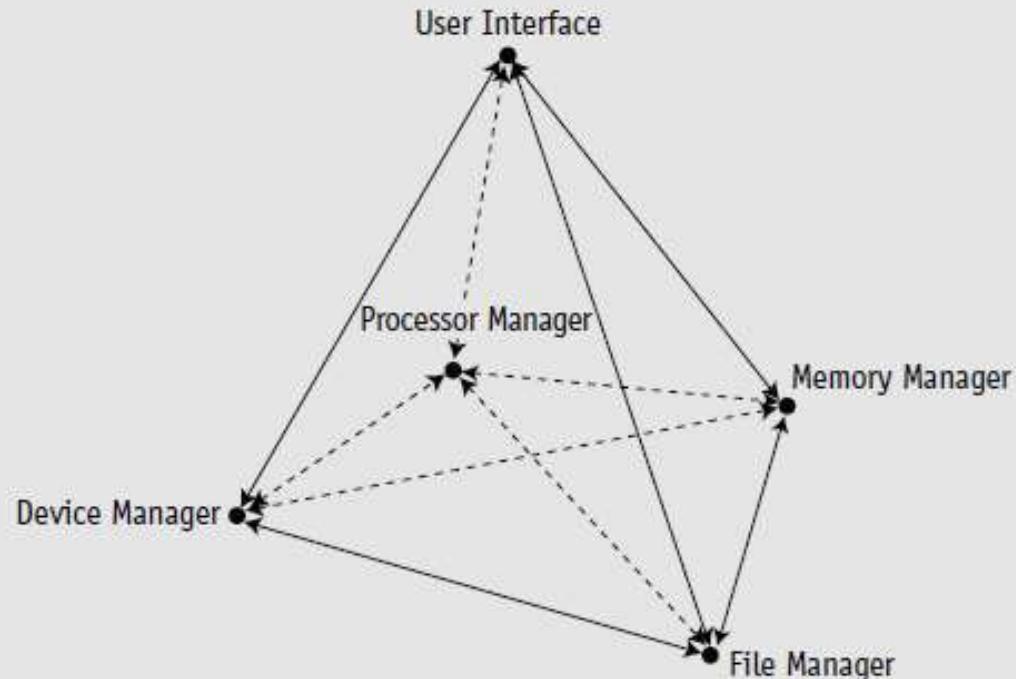
- **Computer System**
 - Software (programs)
 - Hardware (physical machine and electronic components)
- **Operating System**
 - Part of computer system (software)
 - Manages all hardware and software
 - Controls every file, device, section of main memory and nanosecond of processing time
 - Controls *who* can use the system
 - Controls *how* system is used

Operating System Software

- Includes four essential subsystem managers
 - Main Memory Manager, Processor Manager
 - Device Manager and File Manager (hard drives)

(figure 1.1)

This model of a non-networked operating system shows four subsystem managers supporting the User Interface.



Operating System Software (cont'd.)

- Each manager:
 - Works closely with other managers
 - Performs a unique role
- Manager tasks
 - Monitor its resources continuously
 - Enforce policies determining:
 - Who gets what, when, and how much
 - Allocate the resource (when appropriate)
 - Deallocate the resource (when appropriate)

Design Considerations

- Most common overall goal
 - **Maximize use of the system's resources**
 - **Maximise throughput** (memory, processing...)
 - **Minimize downtime**
- Factors included in developmental efforts
 - RAM resources
 - CPUs: number and type available
 - Peripheral devices: variety likely to be connected
 - Networking capability
 - Security requirements, etc.

LMC outputs

Before compile

The screenshot shows the LMC software interface. On the left is a text editor window displaying assembly code. On the right is a message window and control buttons at the bottom.

Messages:

```
Load a Project Clear Compile
00 34 68
01 35 69
02 36 70
03 37 71
04 38 72
05 39 73
06 40 74
07 41 75
08 42 76
09 43 77
10 44 78
11 45 79
12 46 80
13 47 81
14 48 82
15 49 83
16 50 84
17 51 85
18 52 86
19 53 87
20 54 88
21 55 89
22 56 90
23 57 91
24 58 92
25 59 93
26 60 94
27 61 95
28 62 96
29 63 97
30 64 98
31 65 99
32 66 99
33 67 99
```

Control Buttons:

Clear, Reset, Run, Slow, Step, Halt, Counter: 0, MEM Address: 00, Inbox: [Input Field], Enter, Accumulator: 0, MEM Data: 000, Outbox: [Output Field]

After compile and run

The screenshot shows the LMC software interface after compilation and execution. The assembly code has been converted into machine code, and the message window displays various runtime messages.

Messages:

```
Load a Project Clear Compile
00 901 34 68
01 310 35 69
02 901 36 70
03 311 37 71
04 210 38 72
05 808 39 73
06 510 40 74
07 211 41 75
08 902 42 76
09 000 43 77
10 009 44 78
11 007 45 79
12 46 80
13 47 81
14 48 82
15 49 83
16 50 84
17 51 85
18 52 86
19 53 87
20 54 88
21 55 89
22 56 90
23 57 91
24 58 92
25 59 93
26 60 94
27 61 95
28 62 96
29 63 97
30 64 98
31 65 99
32 66 99
33 67 99
```

Control Buttons:

Clear, Reset, Run, Slow, Step, Halt, Counter: 10, MEM Address: 09, Inbox: 7, Enter, Accumulator: 2, MEM Data: 000, Outbox: 002

Runtime Messages:

- COMPILE PROJECT -
- Variable OUTPUT defined as 08
- Variable FIRST defined as 10
- Variable SECOND defined as 11
- INP - 901
- STA - 310
- INP - 901
- STA - 311
- SUB - 210
- BRP - 808
- LDA - 510
- SUB - 211
- OUT - 902
- HLT - 000
- DAT - 000
- DAT - 000
- COMPILE SUCESS
- PROGRAM STARTING -
- 901 - Asking user for input
Text inputed: 9
- 310 - Storing the value of the Accumulator in address: 10
- 901 - Asking user for input
Text inputed: 7
- 311 - Storing the value of the Accumulator in address: 11
- 210 - Subtracting contents of address: 10 to the accumulator, new value: -2
- 808 - The value in the accumulator isn't 0 or positive!
- 510 - Loading the value of address: 10 into the accumulator
- 211 - Subtracting contents of address: 11 to the accumulator, new value: 2
- 902 - Outputting: 002
- 000 - Program Halted

Program control (iteration)

While value = 0 Do
Task(s);
NextStatement (print output)

45	LDA 90	590	90 is assumed to contain value
46	BRZ 48	748	Branch if the value is zero
47	BRA NEXT	660	Exit loop; jump to Next (position 60)
48	.		
	.		
	.		
59	BRA 45	645	End of Task; loop to test again
60	NEXT OUT		Output result

Potential Exam Questions

- What parts of the L.M.C. correspond to the elements of the Von Neumann architecture
(4 marks)
- Describe how the LMC performs the Fetch and execute cycle
(8 marks)

Setting up linux in the labs

- The linux IP address is: 147.252.250.34
 - Use putty to ssh to server/ ssh via cmd window
 - Enter your user-name and password you use on the lab desktops
 - Ubuntu app or
 - Virtual machine (a basic ubtunu VM is on Brightspace)
- Lab 1: Create a file called HelloWorld.c using the vi/nano text editor. (refer to instructions given in the lab section of brightspace)
- Any issues?