

**Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)
Department of CSE
Tutorial-1**

Programme: B.E
Course: Computer Organization

Term: Jan to May 2018
Course Code: CS45

Name: PAVAN D. E	Marks: 19/10	Date:
USN: MS1ECS009	Signature of the Faculty:	<i>D. W. D.</i>

Activity I: Assembling and disassembling of a computer

Objective: To demonstrate the functional units of a system.

Assembling of a system: A PC computer is a modular type of computer, it can be assembled using hardware components made by different manufacturers, so as to have a custom built computer according to one's specific needs.

Disassembling of a system: When referring to hardware, **disassemble** is the process of breaking down a device into separate parts. A device may be disassembled to help determine a problem, to replace a part, or to take the parts and use them in another device or to sell them individually.

Activity to be performed by students: Identify the different parts of the system including its interconnection. Observe the assembly and disassembly procedure.

Answer the following questions.

1. Write down the detailed procedure to assemble a system.
2. Explain how troubleshooting a system helps to trace and correct the faults in a system
3. List out the procedure to install extra memory card to a system
4. With a diagram explain different cables used to connect function units in a system.
5. Discuss the safety precautions one should take while removing components of a system

MARKS :

Name :	PAVAN D G	Branch:	CSE
USN/Roll No. :	IIMS18CS088	Sem/Sec:	TY 'B'
Subject :	CO A	Subject Code:	

1. Write down the detail procedure to assemble the system.

A:- — Before starting assembling the computer system, make sure you have the screws and a screwdriver.

— The first step for assembling the computer system starts with mounting the processor on the processor socket of the motherboard. Once the processor is mounted, the heat sink will be attached on top of the processor. The CPU fan also attached on top of the heat sink.

— The motherboard is to be fixed vertically in the tower case and the screws are fixed from behind of the motherboard.

- Line up the power supply at the top back end of the cabinet and screw it.
- Install the CD/DVD drives at the top front end of the cabinet and screw it. Install the Hard disk drive and floppy disk drive below CD/DVD drive & screw it.
- Select the appropriate data cable to connect one end of the cable to its drive socket and another end at its appropriate connector on the motherboard.
- For SATA hard disk drive (or) CD/DVD drive use SATA cable and its power cable else use IDE data cable.
- Mount the memory modules on the motherboard by aligning the RAM to its socket on motherboard and press it downward.
- Install the internal cards to its socket and attach the cables / power cable to it.

MARKS :

Name :	PAVAN D.G	Branch:	CSE
USN/Roll No. :	IMS18CS088	Sem/Sec:	IV /B1
Subject :	COA	Subject Code:	

- Cover the tower by placing it and pressing towards front side of screen it.
 - Connect the external devices with CPU at its appropriate socket. Monitor at the video output socket.
 - Connect the power cable to the back of tower in SMPS. Plug in power cable to the electric board.
- 2) Explain how trouble shooting a system helps to trace and correct the failure in the system.
- A:- Troubleshooting is a form of problem often applied to repair failed products or processes on a machine (or) a system. It is a logical, systematic search for the sources of a problem in order to solve it and

make the product or process operational again troubleshooting is needed to identify the symptoms. Determining the most likely cause is a process of elimination, eliminating potential causes of a problem. Finally, troubleshooting requires confirmation that the solution restores the product (or) process to its working.

Step 1 :- Identify the problem.

Step 2 :- Establish a theory to determine cause.

Step 3 :- Test the theory to determine cause.

Step 4 :- Establish a plan of action to resolve the problem and implement the solution.

Step 5 :- Verify full system functionality and if applicable implement preventive measures.

Step 6 :- Document findings, actions & outcomes.

3) List out the procedure to install extra memory card to a system.

A:- Step 1 :- Disconnect the power cable from the system and if needed, unplug other back-panel

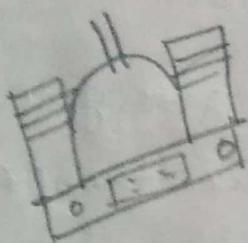
cables so that you can safely turn your system on to its side.

Step 2: Remove the side panel to give you full access to the interior and locate the RAM slots. They are most commonly found next to the processor and its cooler. If there is already RAM in your system, eject it by pressing firmly on the tabs on the motherboard at either end of the slot.

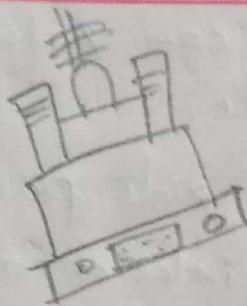
Step 3: To install the new RAM, line up notches in the bottom of the sticks with gaps in the slot on the motherboard.

Step 4: Once the sticks have clicked into place confirm that the wing clips are locked into hold the stick firmly.

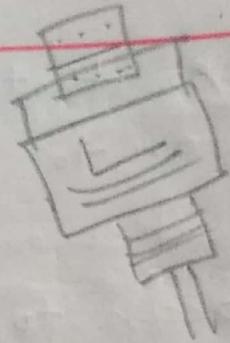
4) With a neat diagram explain different cables used to connect functional units in a system.



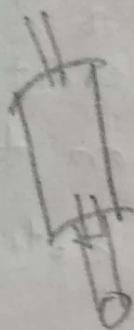
VGA cable



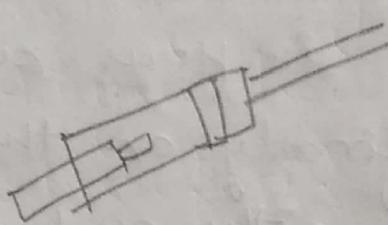
DVI cable



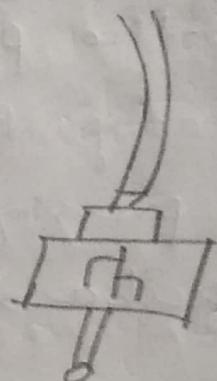
HDMI cable



PS/2 cable



Ethernet cable



USB cable



Computer power cord (iec60320 plug).

VGA cable — connect one end to computer, monitor, TV connect other end to VGA.

DVI cable — connect one end to computer, monitor, other end to DVI part on computer.

HDMI cable :- connect one end to computer monitor & the other end to HDMI port.

PS/2 cable :- connect one end to PS/2 keyboard / PS/2 mouse, other end to PS/2 ports on computer.

Ethernet cable :- connect one end to router / network switch, other end to ethernet port on computer.

USB cables one end to USB device other end to USB ports on computer.

Q) Discuss the safety precautions one should take while removing components of a system.

A :- 1. Fully shutdown and unplug the computer before disassemble the system.

2. Take off any metal objects on your arms such as rings even if your unit is unplugged.

3. Make sure your hands are completely dry.
4. Work in a cool area to avoid perspiration.
5. Before touching any part within the tower put your hands against another metal surface.
6. Prepare a place to keep any screws you may remove.
7. If a component doesn't come out easily, do not forcefully remove it.
8. Be careful when handling the motherboard.
9. Handle all parts with care. Place each piece you remove carefully down onto a stable surface.
10. Take note that the three most damaging things to a computer are moisture, shock & dust.

Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)
Department of CSE

Tutorial -II

Programme: B.E

Course: Computer Organization Course Code: CS45

Term: Jan to May 2020

Name:	PAVAN D.G	Marks:	80 /10	Date:
USN:	1MS18C088	Signature of the Faculty:	CD 4/21/2020	

Activity II: Demonstrating Datapath and instruction execution stages using MarieSim Simulator

Objective: To simulate inter communication between CPU and memory.

Simulator Description: MarieSim is a computer architecture simulator based on the MARIE architecture. It provides users with interactive tools and simulations to help them deepen their understanding of the operation of a simple computer. One can observe how assembly language statements affect the registers and memory of a computer system.

Activity to be performed by students:

1. Draw the interconnection between memory and a processor.

2. List out the steps required to execute an instruction.

3. Write and execute assembly language program to compute

i) $f = (g+h)*(i+y)$

ii) $d = b^2 - 4ac$

4. Describe the factors affecting the performance of a processor

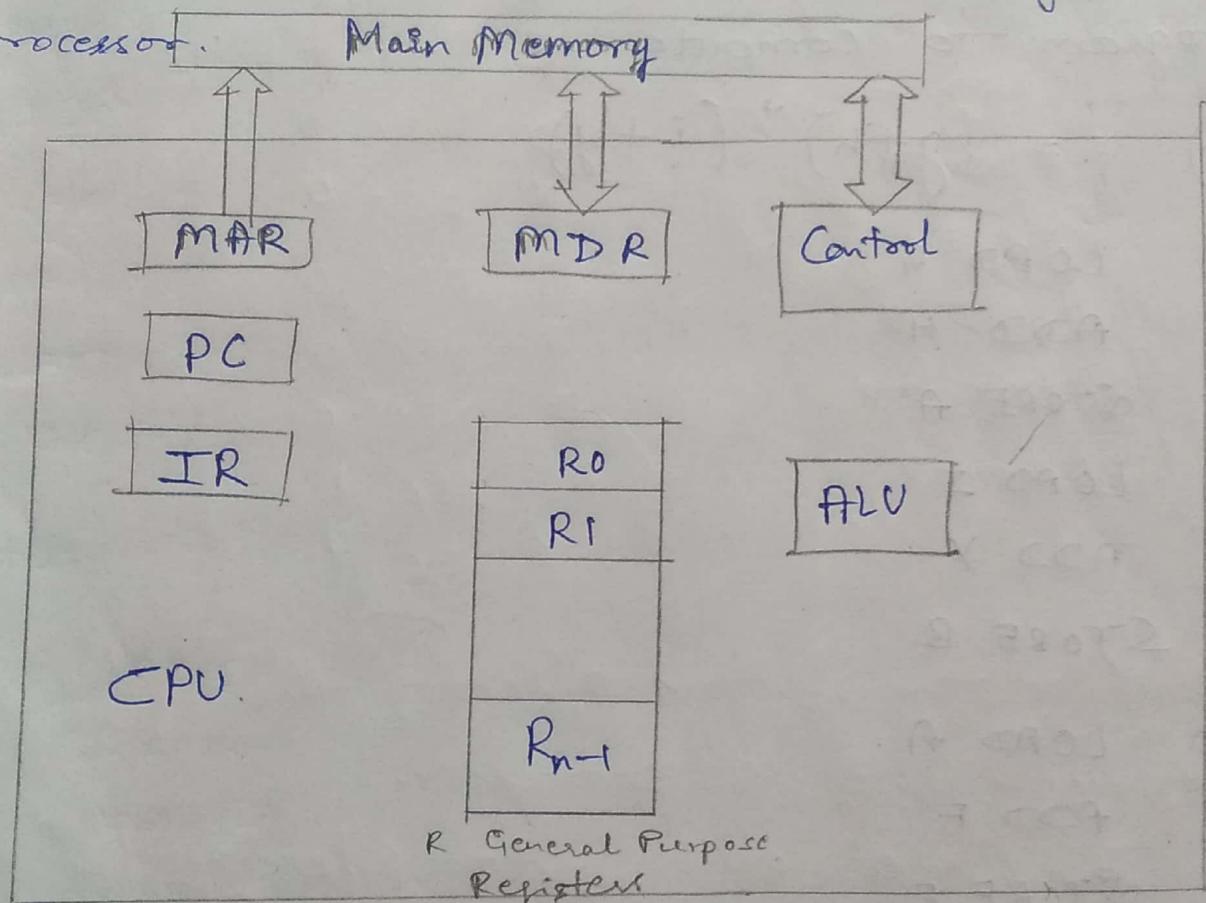
3. Results and Snapshots:



MARKS :

Name:	PAVAN D.G	Branch:	CSE
USN/Roll No.:	1MS18CL008	Sem/Sec:	JVI B)
Subject:	COA	Subject Code:	

- ① Draw the interconnection b/w memory & processor.



- ② List out the steps required to execute an instruction.

A:- There are 6 steps:-

- 1) Fetch instruction
- 2) Decode information.

3) Perform ALU operation.

4) Access memory

5) Update register file

6) Update the program counter (PC)

③ Write and execute the assembly language program to compute.

$$i) f = (g+h)^*(i+y)$$

A:- LOAD G

ADD H

STORE A

LOAD I

ADD Y

STORE B

loop LOAD A

ADD F

STORE F

LOAD B

SUBT ORE

STORE B

JUMP LOOP

LOAD F

Output

HALT

9 DEC 9

H DEC 5

Y DEC 8

I DEC 2

A DEC 0

B DEC 0

F DEC 0

one DEC 1

$$\text{ii) } d = b^2 - 4ac$$

A := LOAD B

STORE 0

LOAD B

ADD X

STORE X

SUBT one

STORE 0

JUMP first

Second

```
LOAD D
ADD Y
STORE Y
LOAD C
SUBT one
STORE C
JUMP second
```

third

```
LOAD four
ADD Z
STORE Z
LOAD Y
SUBT one
STORE Y
Jump third
```

```
LOAD D
ADD X
SUBT Z
OUTPUT
HALT
A DEC 6
B DEC 3
C DEC 2
D DEC 0
X DEC 0
Y DEC 0
Z DEC 0
D DEC 0
one DEC 1
four DEC 4
```

(H) Describe the factors affecting the performance of a processor.

A: There are four factor affecting the performance of a processor.

i) Multiple cores: - Nowadays we have dual, quad and even octa core processors with its own fetch and execute cycles. However, the software should

MARKS :

Name :	PAVAN D.G	Branch:	CSE
USN/Roll No. :	1MS18CS088	Sem/Sec:	IV 'B'
Subject :	COA	Subject Code:	

make use of the multiple cores.

- clock speed - The processor requires a clock pulse in order to operate correctly one clock cycle = 1 Hz A pc clock speed is normally in GHz region.
- Cache memory : It is a small amount of high performance RAM that is built into the processor. This RAM stores the data which has to be repeatedly used by the processor if it doesn't require a request from memory.
- Word length : The no. of bits the CPU can process simultaneously , for ex, a 32-bit processor because of the wider word length.

- Address Bus width :- It is the width of the address bus and determines the maximum amount of addressable locations. For example, an address bus of 8 bit means that you can have 256 addresses (0 to 255)
- + Data Bus width :- It is the no. of bits that can be transferred simultaneously from one device to another. If the data bus is 16 bits and the address bus is 12 bits, so the data is fetched in 2×16 bit groups.

MARIE Assembler Code Editor

File Edit Assemble Help

```
LOAD X  
ADD Y  
STORE Z  
HALT  
X, DEC 5  
Y, DEC 1  
Z, DEC 0
```

D:\CD\Lab\MARIE\Datapath_Simulators\add.mas Assembly successful



Type here to search



M Assembly Listing for add.mas

Assembly listing for: add.mas

Assembled: Tue Jan 28 06:57:27 IST 2020

```
000 1004 I      LOAD X
001 3005 I      ADD Y
002 2006 I      STORE Z
003 7000 I      HALT
004 0005 I  X    DEC 5
005 0001 I  Y    DEC 1*
006 0000 I  Z    DEC 0
```

Assembly successful.

SYMBOL TABLE

Symbol	Defined	References
--------	---------	------------

X	I 004	I 000
Y	I 005	I 001
Z	I 006	I 002

Print

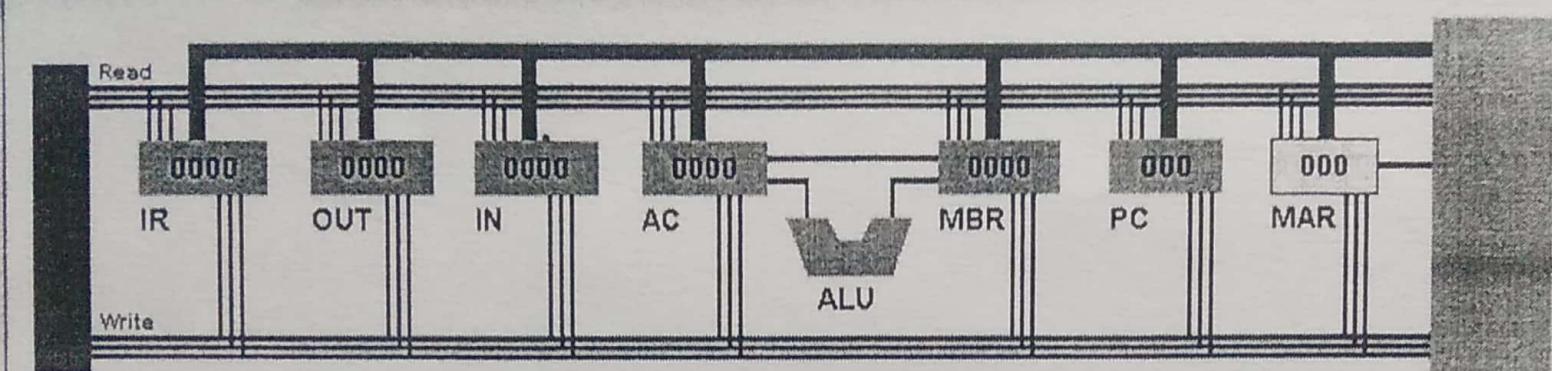


Type here to search



MARIE Data Path Simulator

File Stop Step Set Speed Sgt Fast Fetch On Help



Control Unit

(Fetch cycle) MAR <-- PC

Main Memory

	label	opcode	operand	hex				
000		LOAD	X	1004				
001		ADD	Y	3005				
002		STORE	Z	2006				
003		HALT		7000				
004	X	DEC	5	0005				
005	Y	DEC	1	0001				
006	Z	DEC	0	0000				

Trace

Print

Input

Hex

IR OUT IN AC MBR PC MAR

IR OUT IN AC MBR PC MAR

0000 0000 0000 0000 0000 000 000

OS (C:
DATA (D:
Network

13 items



Type here to search



label opcode operand hex

000	LOAD	X	1004
001	ADD	Y	3005
002	STORE	Z	2006
003	HALT		7000
004	X	DEC	5
005	Y	DEC	1
006	Z	DEC	0

AC 0006 (Hex)
IR 3005 (Hex)
MAR 005 (Hex)
MBR 0001 (Hex)
PC 002 (Hex)
INPUT ASCII

OUTPUT

ASCI Control

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
000	1004	3006	2006	7000	0005	0001	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
020	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
030	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
070	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
080	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Press [Step] to continue.

label	opcode	operand	hex
000	LOAD	X	1004
001	ADD	Y	3005
002	STORE	Z	2006
003	HALT		7009
004	X	DEC	5
005	Y	DEC	1
006	Z	DEC	0

AC 0006 (Hex)

IR 7000 (Hex)

MAR 003 (Hex)

MBR 0006 (Hex)

PC 004 (Hex)

INPUT ASCII ▾

OUTPUT

+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	1004	3005	2006	7000	0005	0001	0006	0000	0000	0000	0000	0000	0000	0000	0000
010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
020	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
030	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
070	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
080	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Machine halted normally.

Tutorial -III

Programme: B.E

Term: Jan to May 2020

Course: Computer Organization Course Code: CS45

Name: PAVAN D.G	Marks: 10/10	Date: 04/02/20
USN: 1M18CS088	Signature of the Faculty:	<i>[Signature]</i>

Objective: To simulate ARM Instruction set using ARMSim simulator.

Simulator Used: ARMSim 1.91 is a desktop application running in a Windows environment. It allows users to simulate the execution of ARM assembly language programs on a system based on the ARM7TDMI processor.

ARM enables the users both to debug ARM assembly programs and to monitor the state of the system while a program executes.

Activity to be performed by students:

- 1) Write an ARM program to perform basic arithmetic operations.
- 2) Write an ARM program to demonstrate the working of load and store instructions.
- 3) Write an ARM program to evaluate expression $f=(g+h)-(i+j)$
- 4) Write an ARM program to find the sum of all elements of an array.
- 5) Write an ARM program to find the factorial of a number.

Programs and the snapshots:

MARKS :

Name :	PAVAN D.G	Branch:	CSE
USN/Roll No. :	1MG18CS088	Sem/Sec:	IV/B
Subject :	COA	Subject Code:	

Q) Write a ARM program for arithmetic operations.

A:-

```

    MOV R5, #10
    MOV R6, #20
    ADD R7, R5, R6
    SWI 0x11
  
```

a) Write an ARM program to demonstrate the working of load & store instructions

A:-

```

    MOV R1, #0x00000040
    MOV R1, #0
    MOV RH, #50
    STR R4, [R1, R2]
    LDR R6, [R1, R2]
    SWI 0x11
  
```

3) mov R1, #20
mov R2, #30
ADD R3, R1, R2
MOV R4, #40
MOV R5, #50
ADD R6, R4, R5
SUB R7, R3, R6
SWI 0x11

4) mov R0, #5
LDR R1, =array
loop LDR R2, [R1], #4
ADD R3, R1, R2
SUB R0, R0, #1
CMP R0, #0
BNE loop

array DCD 0x00000001, 0x00000002, 0x00000003,
 0x00000004

SWI 0x11

5) mov r0, #3
mov r1, r0
mov r2, #1
mov r3, #1

fact

mul r2, r1, r2

sub r1, r1, r3

cmp r1, #1

bge fact

SWI 0x11

```
ADD - Notepad  
File Edit Format View Help  
MOV R5, #10  
MOV R6, #20  
ADD R7, R5, R6  
  
SWI 0x11
```

Activate Windows
Go to Settings to activate Windows.

Type here to search

La 1, Col 1 100% Windows (CRLF) UTF-8
11:36 AM 04-02-2020

ADD.ASM

ARMSim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

Registers

General Purpose	Hexadecimal	Unsigned Decimal	Signed Decimal
R0 : 0	: 0	: 0	: 0
R1 : 0	: 0	: 0	: 0
R2 : 0	: 0	: 0	: 0
R3 : 0	: 0	: 0	: 0
R4 : 0	: 0	: 0	: 0
R5 : 10	: 10	: 10	: 10
R6 : 20	: 20	: 20	: 20
R7 : 30	: 30	: 30	: 30
R8 : 0	: 0	: 0	: 0
R9 : 0	: 0	: 0	: 0
R10 (s1) : 0	: 0	: 0	: 0
R11 (fp) : 0	: 0	: 0	: 0
R12 (ip) : 0	: 0	: 0	: 0
R13 (sp) : 21504	: 21504	: 21504	: 21504
R14 (lr) : 0	: 0	: 0	: 0
R15 (pc) : 4300	: 4300	: 4300	: 4300

ARM Registers

Negative (N) : 0
Zero (Z) : 1
Carry (C) : 0
Overflow (V) : 0
IRQ Disable: 1
FIQ Disable: 1
Thumb (T) : 0
FPU Mode : 8By

Memory

0x0001000C : R00000011 SWI 0x11

OutputView

Console Stdin/Stdout/Stderr

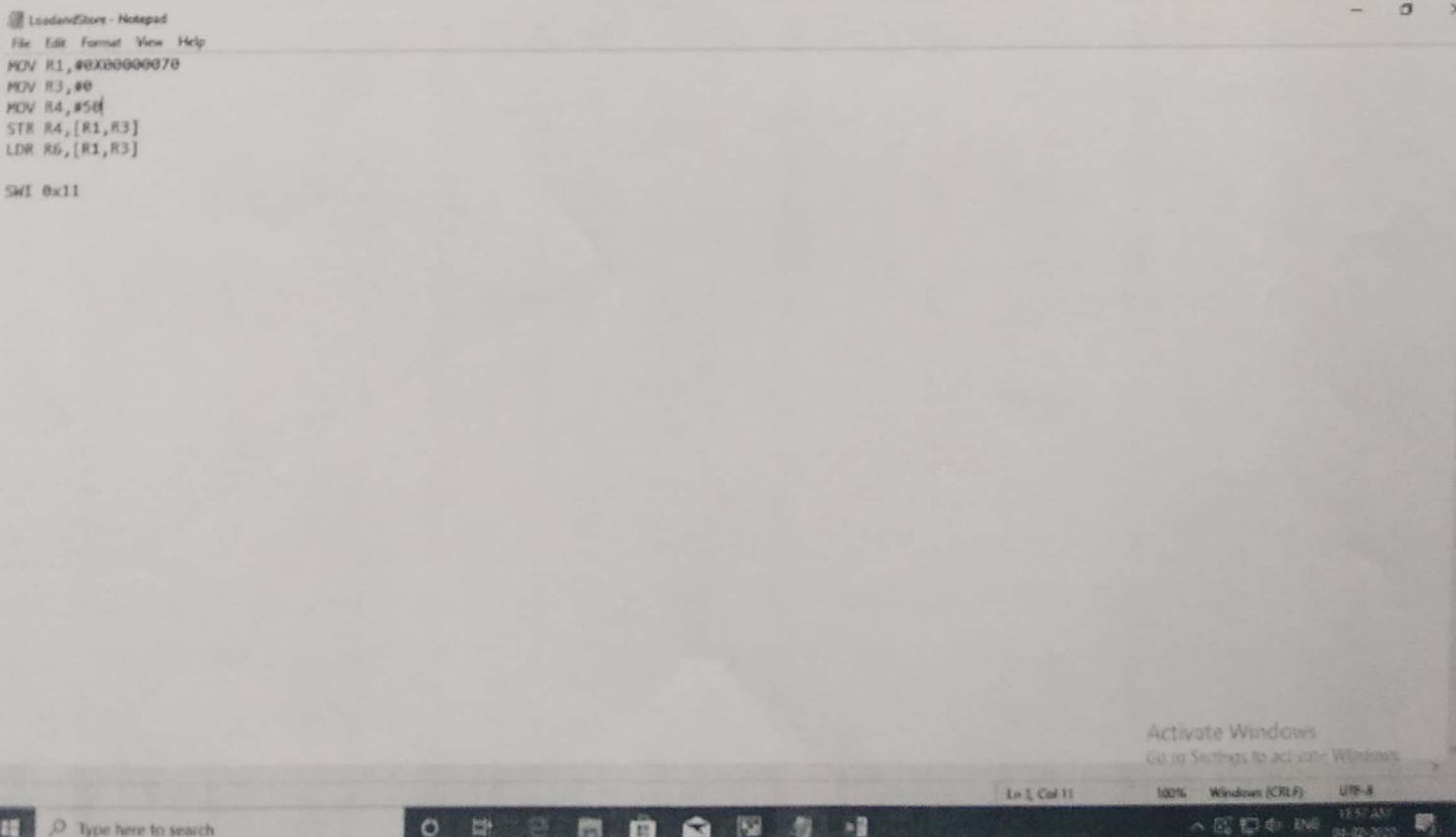
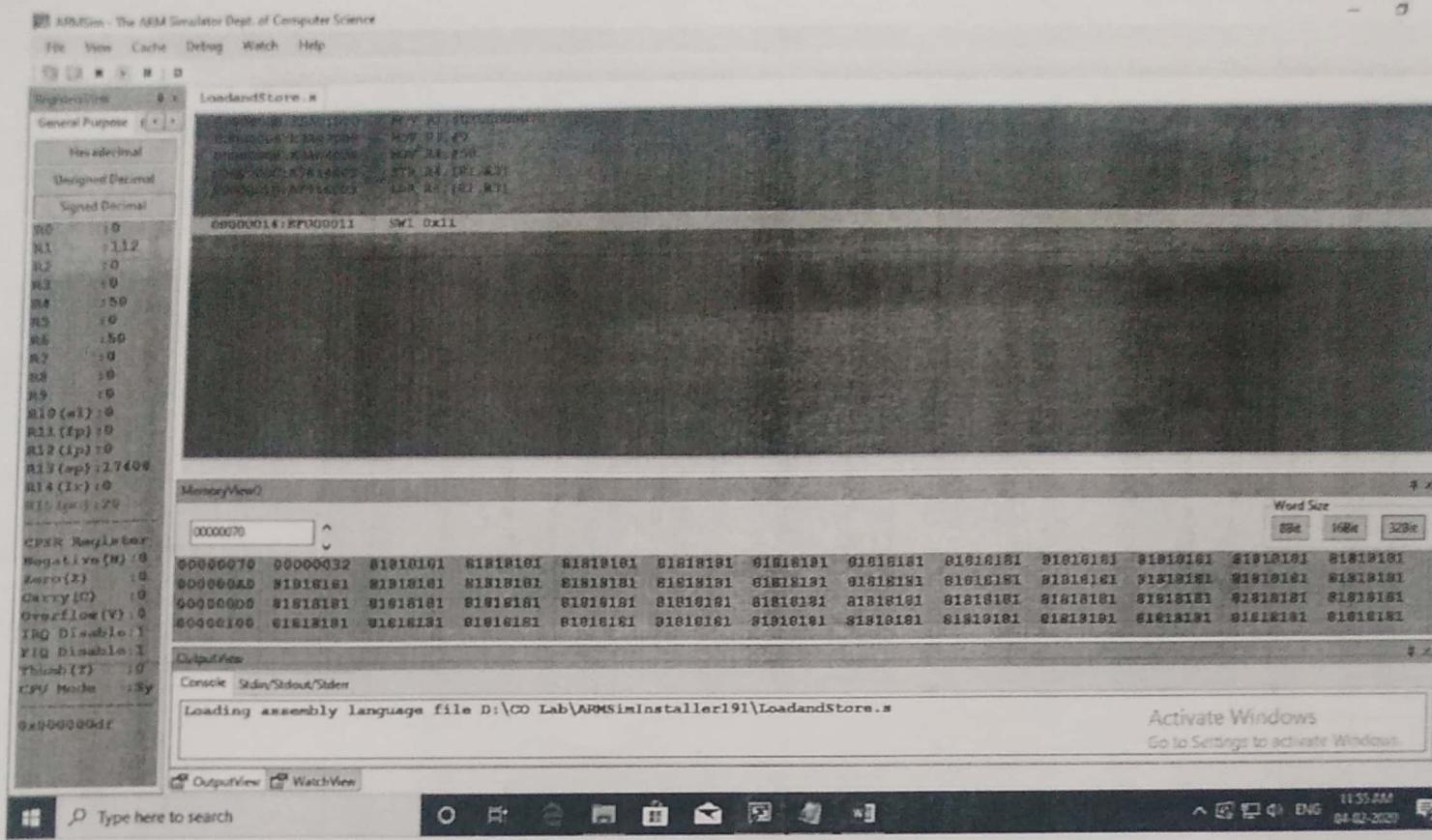
Execution ended. Instruction Count: 0 Elapsed Time: 00:00:00.0468738 Instructions per second: 0

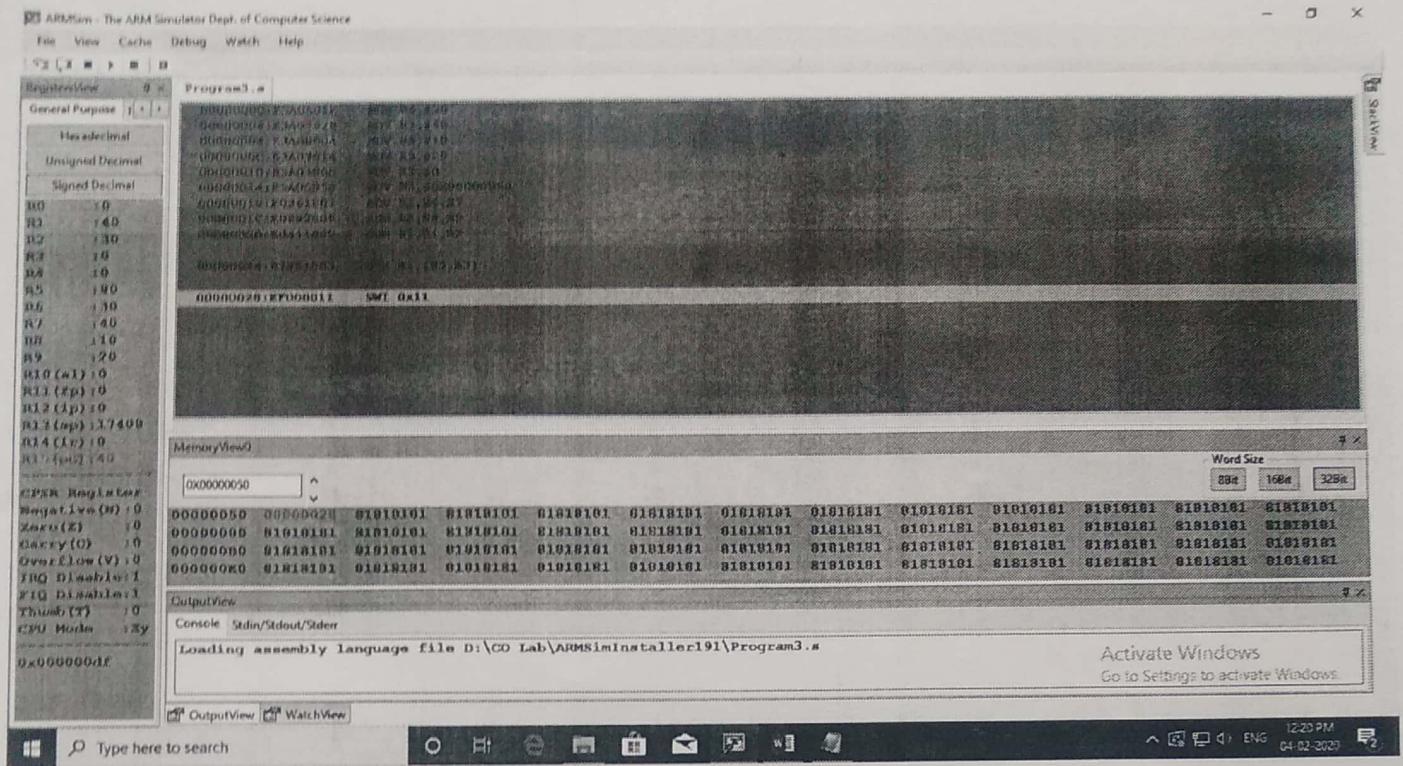
Activate Windows
Go to Settings to activate Windows.

Type here to search

La 1, Col 1 100% Windows (CRLF) UTF-8
11:36 AM 04-02-2020

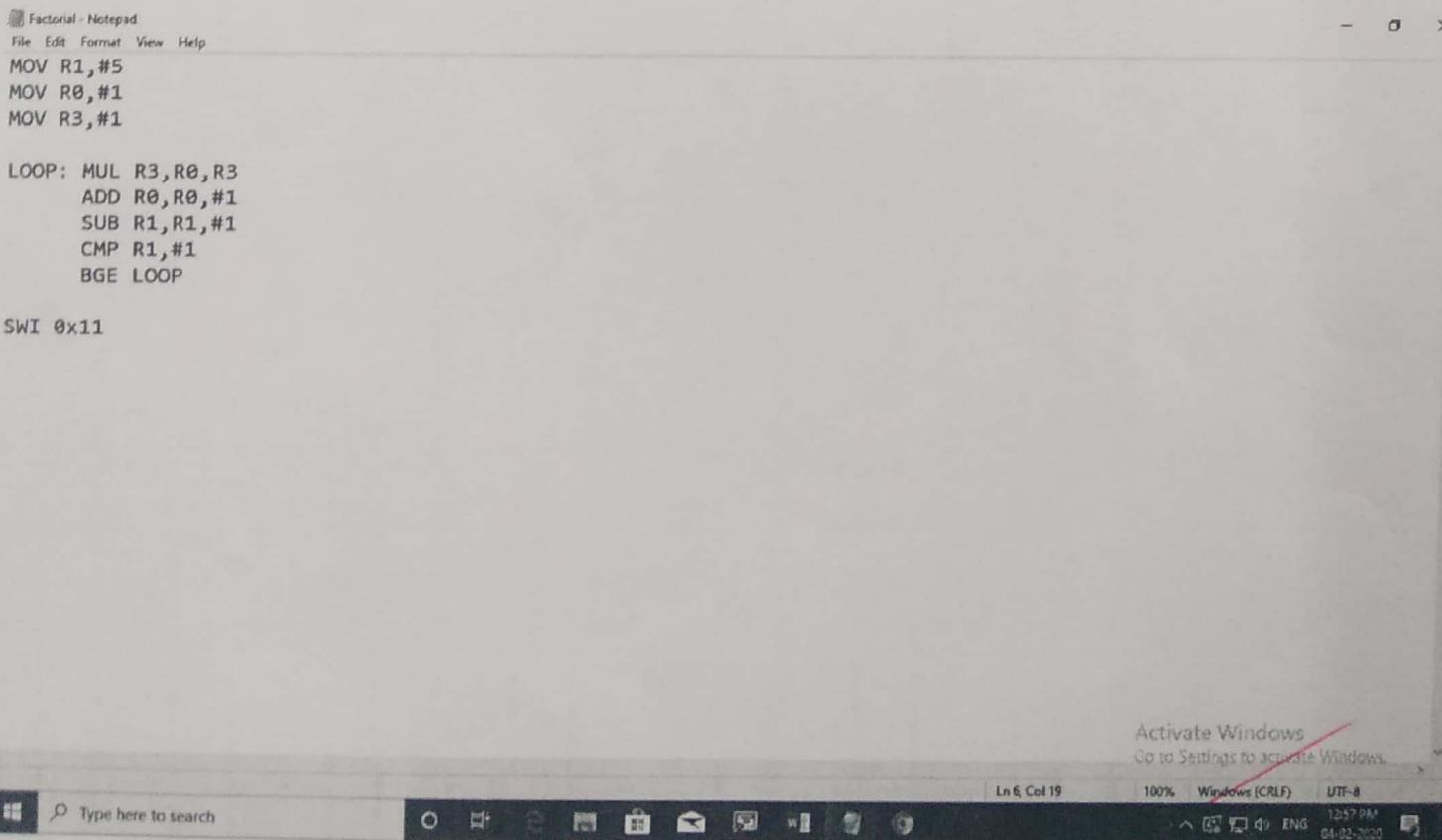
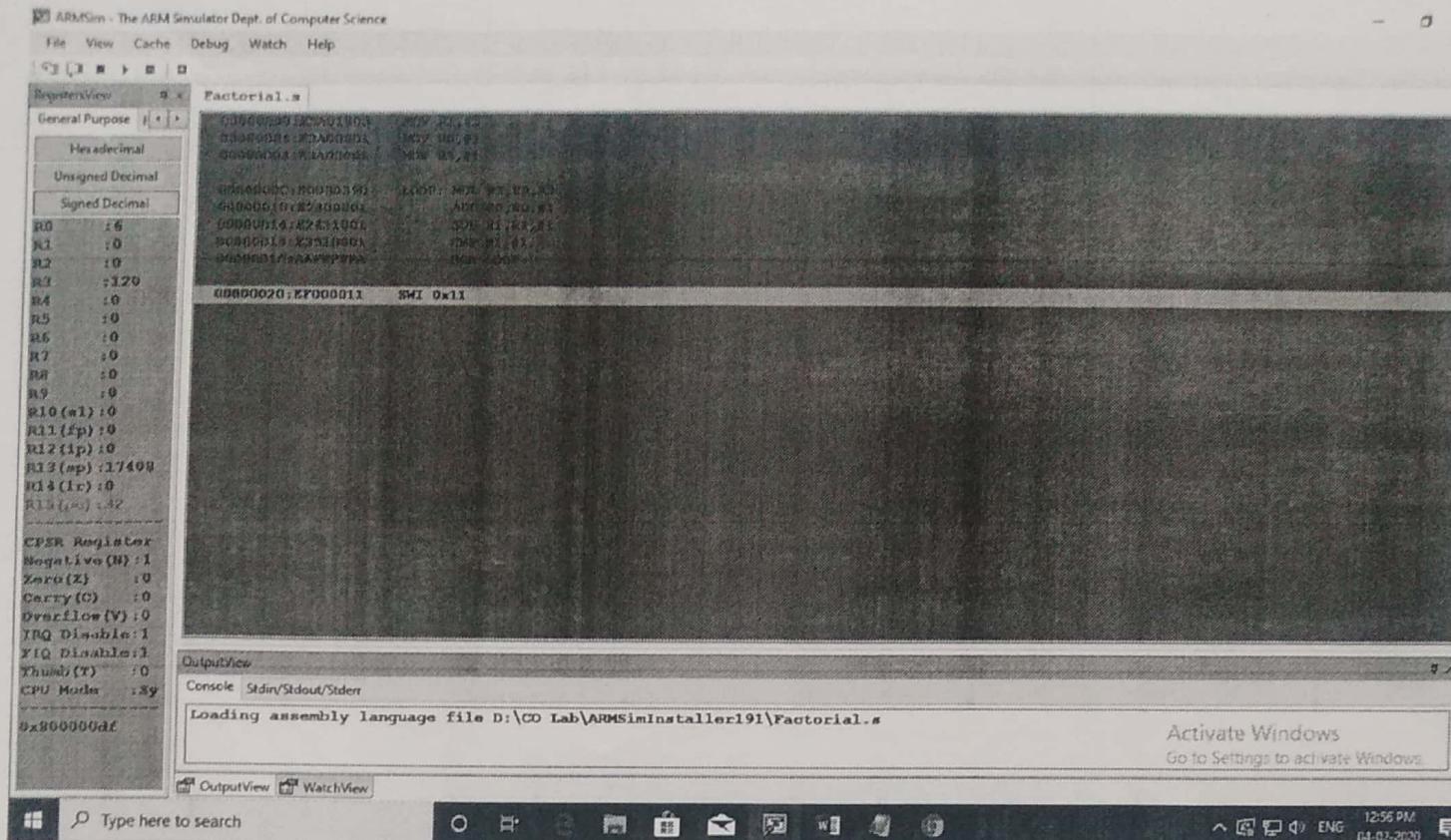
Program 2





```
Program3 - Notepad
File Edit Format View Help
MOV R6, #30
MOV R7, #40
MOV R8, #10
MOV R9, #20
MOV R3, #0
MOV R5, #0X00000050
ADD R1, R6, R7
ADD R2, R8, R9
SUB R1, R1, R2
STR R1, [R5, R3]
SWI 0x11
```

Factorial



Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)

Department of CSE

Programme: B.E
Course: Computer Organization

Term: Jan to May 2019
Course Code: CS45

Activity IV: Executing ARM programs using ARMsim simulator.

Name: PAVAN D.G	Marks: /10	Date: 25/02/20
USN: 1MS18CS028	Signature of the Faculty:	

Objective: To simulate ARM Instruction set using ARMsim simulator.

Simulator Used: ARMSim 1.91 is a desktop application running in a Windows environment. It allows users to simulate the execution of ARM assembly language programs on a system based on the ARM7TDMI processor.

ARM enables the users both to debug ARM assembly programs and to monitor the state of the system while a program executes.

Activity to be performed by students:

- 1) Write an ARM program to generate Fibonacci Series.
- 2) Write an ARM to search an element in an array and print Y if found and print N if not found.
- 3) Write an ARM program to find the length of a string and copying one string to another.



MARKS :

Name :	PAVAN D-6	Branch:	CSE
USN/Roll No.:	1ME18LC088	Sem/Sec:	IV /B'
Subject:	CFA	Subject Code:	CS415

D) MOV R0, #0
MOV R1, #1
MOV R2, #5
MOV R3, #0
MOV R4, = 0X00002000
MOV R5, #0

loop:

STR R0, [R4, R5]
ADD R6, [R0, R1]
MOV R0, R1
MOV R1, R6
ADD R5, R5, #4
ADD R3, R3, #1
CMP R3, R2
BLT loop

SWI 0x11

②

```
Mov R0, #1  
Mov R1, #100  
Mov RH, #25  
LDR R0, = 0x00002000  
Mov RS, #1  
loop LDR, R2, [R0, RS]  
SUB R0, R0, #1  
ADD RS, RS, #1  
CMP R1, R2  
BEQ printy  
CMP R0, #0  
BEQ printbc  
BNE loop
```

```
printx: STR RH, [R0]  
LDR R0, [R0]  
SWI 0x00  
B END
```

Printy :
 CTR R7,[R0]
 LDR R0,[R0]
 SWI 0x00

end SWI 0x11

- 3)
• cgu SWI-open, 0x66
• cgu SWI-close, 0x68
• cgu SWI-print, 0x6b
• cgu SWI-RdInt, 0x6c
• cgu Stdout, 1
• cgu SWI-PrStr, 0x69
• cgu SWI-Exit, 0x11

global-start

text

LDR R0, =filename
mov R1, #0
SWI SWIopen
bcs Exit
mov R9, R0
mov R5, #0

loop start:

Mov r5, #stdout

Mov r0, r9

LDR r8, =Array

bcs afterloop

Str r0, [r8, r5]

Add r5, r5, #4

Mov r1, r0

Mov r0, #stdout

Str SWI - print

Add r4, r4, #1

afterloop:

Mov r5, #20

loop: LDR r2, [r8, r5]

SUB r4, r4, #1

SUB r5, r5, #4

Mov r1, r2

bcr end

bcr loop

end: Mov r0, r9

SWI close

Exit:

swi SWI Exit.

ARMsim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView

General Purpose	Floating Point
Hexadecimal	
Unsigned Decimal	
Signed Decimal	
R0 : 00000003	
R1 : 00000065	
R2 : 00000041	
R3 : 00000000	
R4 : 00000000	
R5 : ffffffc0	
R6 : 00000000	
R7 : 00000001	
R8 : 000000a0	
R9 : 00000003	
R10 (s1) : 00000000	
R11 (fp) : 00000000	
R12 (ip) : 00000000	
R13 (sp) : 00004400	
R14 (lr) : 00000000	
R15 (pc) : 0000000c	

CPSR Register

Negative (N) : 0
Zero (Z) : 1
Carry (C) : 0
Overflow (V) : 0
IRQ Disable:1
FIQ Disable:1
Thumb(T) : 0
CPU Mode :System

0x400000df

reverse.s

```
00000005C:E2444001    sub r4,r4,#1
00000060:E2455004    sub r5,r5,#4
00000064:E1A01002    mov r1,r2
00000068:E3A00001    mov r0,$Stdout
0000006C:EF000065    swi SWI_Printf
00000070:E3A010C5    ldr r1,-NewLine @moves to a new line
00000074:EF000069    swi SWI_PrStr
00000078:E3540000    cmp r4, #0
0000007C:0A000000    beq end
00000080:1AFFFFF4    bne loop

00000084:E1A00009    end:   mov r0,r9 @moving filehandle back to the register where armasm will be expecting it (look at page 21-22 of the manual)
00000088:EF000068    swi SWI_Close @closes the file

0000008C:             Exit:
0000008C:EF000011    swi SWI_Exit @stop execution (must be at the end of all assembly programs)

000000AD:             .data
000000AD:             Array:
000000AD:             .align

000000AD:             FileName: .asciz "input.txt" @this is creating a string with the name of the file
000000AA:             InFileError: .asciz "Unable to open input file\n"
000000C5:             NewLine: .asciz "\n" @creating a string
000000C5:             .end

MemoryView0
```

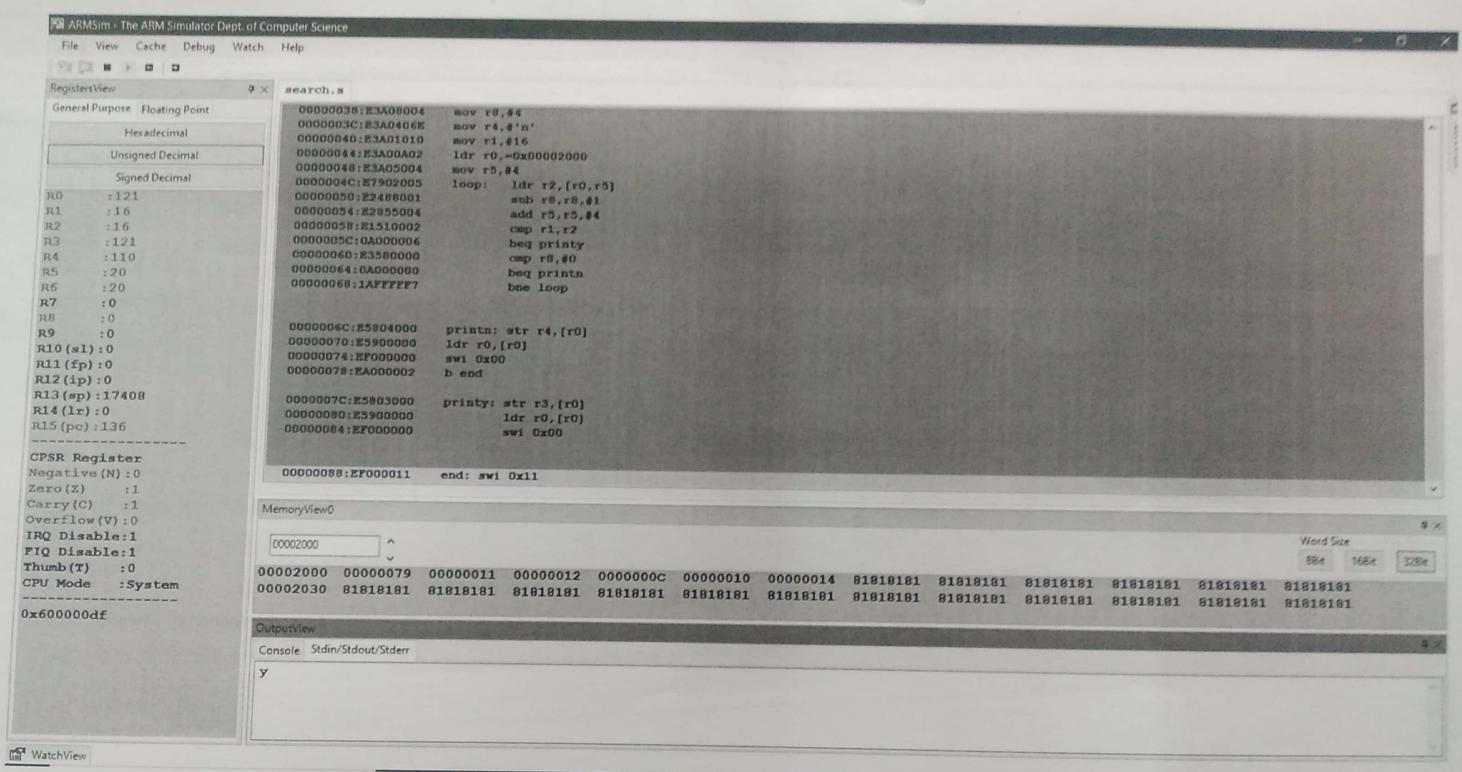
Word Size: 8Bit 16Bit 32Bit

00000034	^
00000034 E1A01000 E3A00001 EF00006B E2844001 E3A00001 E3A010C5 EF000069 EAFFFFF0 E3A05014 E7982005 E2444001 E2455004	
00000064 E1A01002 E3A00001 EF00006B E3A010C5 EF000069 E3540000 0A000000 1AFFFFF4 E1A00009 EF000068 EF000011 00000000	

OutputView

Console Stdin/Stdout/Stderr

70
69
68
67
66



ARMsim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView fibon.s

General Purpose Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 00001a6d R0 : E3A00000

R1 : 00002ac2 R1 : E3A01001

R2 : 00000014 R2 : E3A02014

R3 : 00000014 R3 : E3A03000

R4 : 00002000 R4 : E3A04A02

R5 : 00000050 R5 : E3A05000

R6 : 00002ac2 R6 : E3A06005

R7 : 00000000 R7 : E3A07001

R8 : 00000000 R8 : E3A08001

R9 : 00000000 R9 : E3A09001

R10 (s1) : 00000000 R10 : E3A0A001

R11 (fp) : 00000000 R11 : E3A0B001

R12 (ip) : 00000000 R12 : E3A0C001

R13 (sp) : 00004400 R13 : E3A0D006

R14 (lr) : 00000000 R14 : E3A0E006

R15 (pc) : 0000003c R15 : E3A0F0005

CPSR Register

Negative (N) : 0

Zero (Z) : 1

Carry (C) : 1

Overflow (V) : 0

IRQ Disable: 1

FIQ Disable: 1

Thumb (T) : 0

CPU Mode : System

0x600000df

00000000:E3A00000 mov r0,#0

00000004:E3A01001 mov r1,#1

00000008:E3A02014 mov r2,#20

0000000C:E3A03000 mov r3,#0

00000010:E3A04A02 ldr r4,-0x00002000

00000014:E3A05000 mov r5,#0

00000018:E3A06005 loop: str r0,[r4,r5]

0000001C:E3A07001 add r6,r0,r1

00000020:E3A08001 mov r0,r1

00000024:E3A09006 mov r1,r6

00000028:E3A0A006 add r5,r5,#4

0000002C:E3A0B001 add r3,r3,#1

00000030:E3A0C001 CMP r3,r2

00000034:BAFFFFF7 blt loop

00000038:EF000022 swi 0x22

0000003C:EF000011 swi 0x11

MemoryView0

Word Size: 8Bit 16Bit 32Bit

00002000 00000000 00000001 00000001 00000002 00000003 00000005 00000008 0000000D 00000015 00000022 00000037 00000059

00002030 00000090 000000E9 00000179 00000262 000003DB 0000063D 00000A18 00001055 81818181 81818181 81818181 81818181

00002060 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

00002090 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

OutputView

Console Stdin/Stdout/Stderr

Execution ending, Instruction Count: 0 Elapsed Time: 00:00:00.0080842

Instructions per second: 0

OutputView WatchView

Activity - V

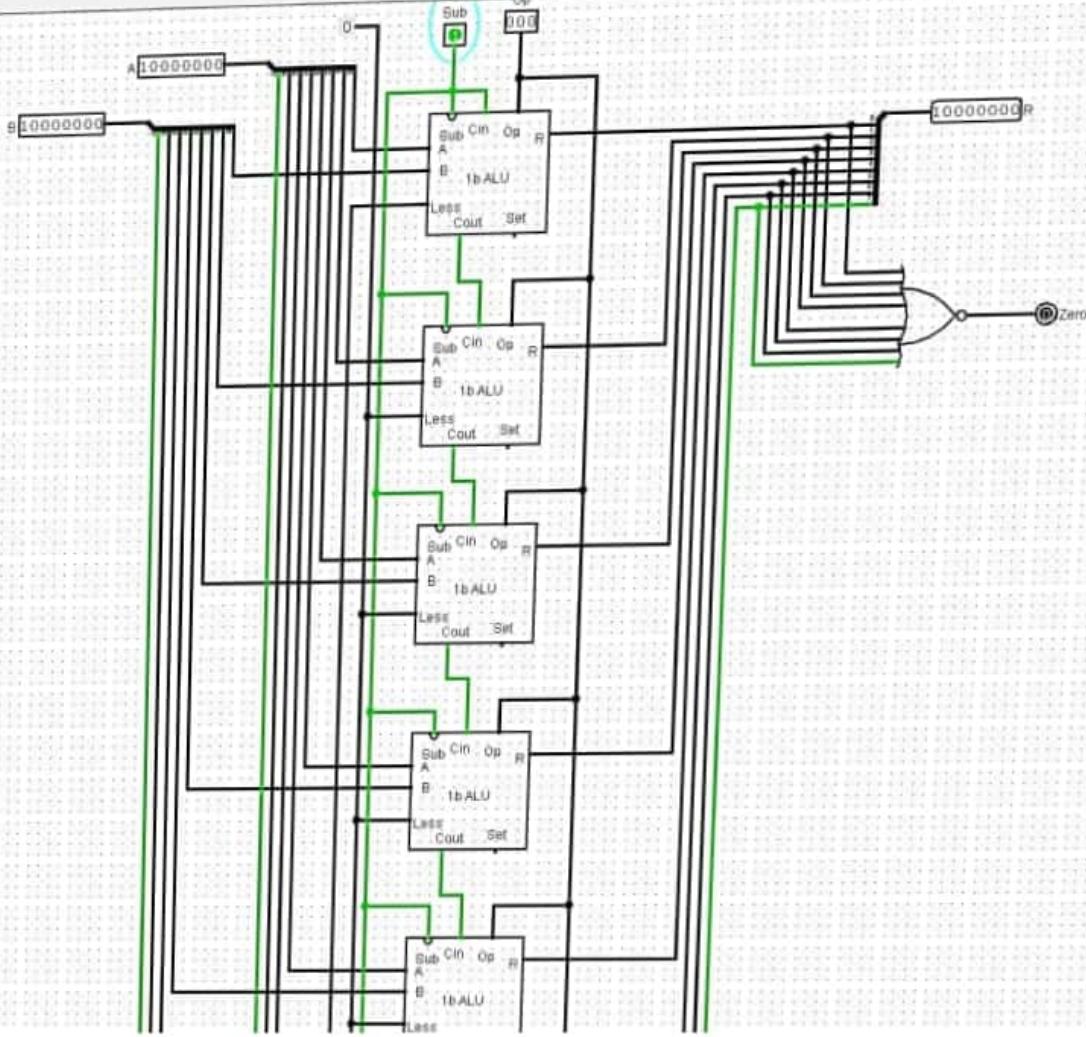
Objective: To simulate the working of Arithmetic and logic unit using Simulator.

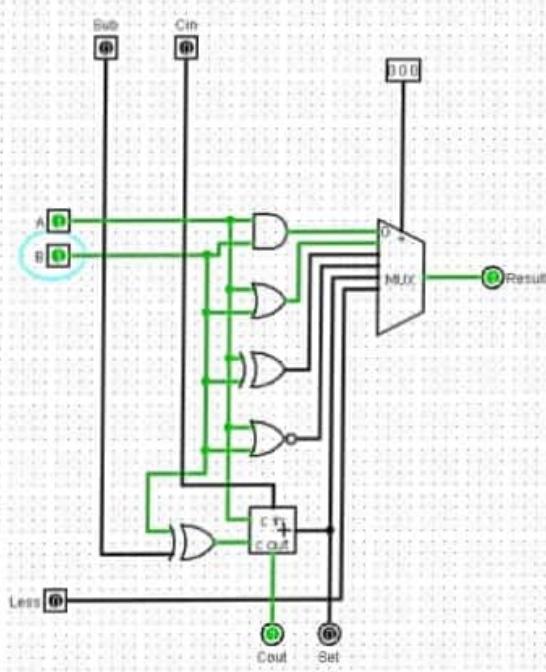
Simulator Description: Logism is an educational tool for designing and simulating digital logic circuits, with its simple toolbar interface and simulation of circuit as you build them it is simple enough to facilitate learning the most basic concepts related to logic circuits with other capacity to build large circuits from smaller sub circuits and to draw bundles of wires with a single mouse click, logism can be used to design and simulate entire CPU's for educational purposes.

Steps in designing ALU:

- Add the two input pins name them A & B
- Add or, and, xor, nor gates and a 1-bit adder.
- Connect the A's & B's of all the gates to their respective pins.
- Add an output pin and name it result.

5. Add a 1-bit multiplexer with 3 select bits.
6. Connect outputs of all the gates to the mux.
7. Connect 3-bit input pin to mux.
8. Add i/p pin to cin, and output pin to cout.
9. Add an ~~X~~-or gate connect its o/p to cout. The first i/p must be connected to B and the second to another i/p pin sub.
10. Add another i/p and name it len. Connect it to the mux.
11. Add an output pin and name it set, connect it to the o/p of add unit.





Computer organisation & Architecture

Activity VI

Parav D.G
(MS186008)

Objective: To simulate the writing operation on memory.

Simulator description: Logism is an educational tool for designing and simulating digital logic circuits with its simple toolbar learning the most basic concepts related to logic circuits with the capacity to build larger circuits from smaller sub circuits and to draw bundles of wires with a single mouse drag, Logism can be used to design and simulate entire CPU's for educational purposes.

The steps in designing memory system:-

1. Add a RAM with separate load and store selected.
2. Add a counter and connect Q to A of the RAM.
3. Add a controller buffer and connect its o/p to the RAM.

4. Add a clock and connect to the ifp of the buffer.
5. Add a TTH unit with 32 rows and columns. Make the connections with RAM.
6. Add a 7-bit random number generator, connect Q to D.
7. Add another connected buffer, connect it to TTH. Also add an if pin to the buffer.
8. Connect the o/p of the second buffer to the counter.
9. Connect a button to the counter.

