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: Keerthana Ningaraju	**Marks:** 10/10	**Date:** 21/01/2020
USN: 1MS18CS059	**Signature of the Faculty:**	4/2/2020

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

RAMAIAH
Institute of Technology

Data Sheet

MARKS :

Name :	Keerthana Ningaraju	Branch:	CSE
USN/Roll No. :	1MS18CS059	Sem/Sec:	IV 'B'
Subject :	Computer Organization Lab	Subject Code:	

→ Write Down the detailed procedure to assemble a system.

Step1: Remove side panels on case. After removing the case from the box. The panels removing from this case with thumb screws.

Step 2: Insert motherboard - Before acting the board in, the I/O panel faceplate needs to be snapped into the location in the back of the case, into the location. Once the board is resting in the case, line up the first hole.

Step3: Check clearances - Being that the computer includes high performance components, some of them are large enough that clearance can become an issue

Step 4: Front Panel Connections - Attach the connections for the buttons, lights, USB ports and audio communication

Step 5: Install Power Supply - Cables that are needed are plugged into the unit

Step 6: Power motherboard - The largest motherboard power cable is to be connected

Step 7: Installing Optical Drive - The optical drive for the computer is a DVD/CD read/write combo

Step 8: Installing the hard drives - Computer uses 4 drives, two in raid and the rest for a main drive and miscellaneous storage

Step 9: Connect Cables - Connect the cables the hard drives and opticals drives. The cables are keyed so they will only be in one direction into the board.

Step10: Installing RAM - The slots are keyed as are the RAM sticks, so make sure the notch is lined up

Step11: Install graphics cards and expand cards - the network card and audio card for the computer and connect into the slots below the graphics card.

Step12: Cable Management:- Organization and hiding cables for high airflow and security / safety

---

Q. Explain how troubleshooting a system helps to trace and correct the false in the system.

A. Troubleshooting is a form of problem solving often applied to repair failed products or processes on a machine or a system. It is a logical, systematic search for the source 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 of probable 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 the full system functionality and if applicable implement preventive measure

Step 6: Document findings, actions and 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

apply 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 access to the interior and locate the RAM slots. They're most commonly found next to the processor and its cooler. If there already RAM in your system, eject it by pressingly firmly on the tabs on the motherboard at either end of the slots

Step 3: To install the new RAM, line up the notches in the bottom of the sticks with the gaps in the slot on the motherboard. Make sure the wings at either end of the slots are pushed back, so that they are tilted away from the RAM. As it does the wings will clamp in and hold the memory securely

Step 4: Once the sticks have clicked into it confirm that the wing clips are locked in to hold the sticks firmly in their slots and then close the PC back up. Plug all the cables back in and try to boot the system

---

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

A. 1. VGA Cable:

Also known as D-sub cable, analog video cable. Connect one end to computer monitor, television
Connect other end to: VGA port on computer

2. DVI Cable:

Connect one end to computer monitor. Connect other end to DVI port on computer.

3. HDMI Cable:

Connect one end to computer monitor, television. Connect other

end to computer monitor, television. Connect other end to HDMI port on computer.

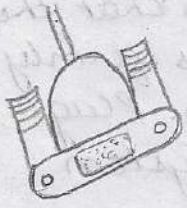
4. PS/2 Cable:- Connect one end to PS/2 keyboard, PS/2 mouse. Connect other end to PS/2 ports on computer. Purple PS/2 port: keyboard. Green PS/2 port: mouse.

5. Ethernet code:- Connect one end to router, network switch. Connect other end to ethernet port on computer

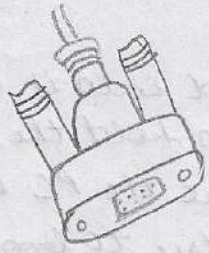
6. USB cable:- connect one end to USB ~~disitt~~ device. Connect other end to USB ports on computer

7. Computer Power Cord (Kettle plug): Connect one end to AC power socket. Connect other end to power supply unit, computer ~~no~~ monitor

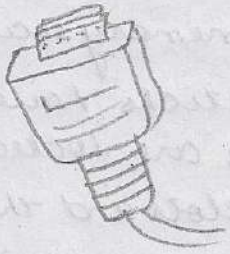
Diagram:



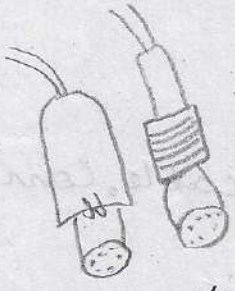
1). VGA Cable



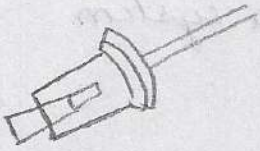
2) DVI cable



3). HDMI Cable

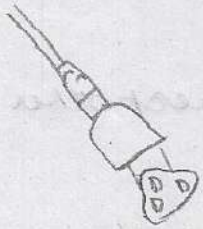


6). PS/2 cable



4). Ethernet Cable

5) USB Cable



7). Computer Power Cord (Kettle Plug)

Data Sheet

RAMAIAH
Institute of Technology

MARKS :

Name :	Kavithara Ningaraju	Branch:	CSE
USN/Roll No. :	1MS18CS059	Sem/Sec:	IV 'B'
Subject :	COLab	Subject Code:	

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

A A few warnings and reminder before you start disambling your computer tower to keep both your unit and yourself safe :

1. Fully shutdown and unplug the computer before you make any attempts to disassemble the tower
2. Take off any metal objects, on your arms or rings such as bracelets, rings or watches.
3. make sure your hands are completely dry to avoid damaging any mechanical parts as well as to avoid electrocution
4. Work in cool areas to avoid perspiration for the same person as sun in the ~~por~~ previous number
5. Before touching any part within the tower, put your hands against another metal surface to remove static charge, which may damage sensitive devices.
6. Prepare a place to keep any screws you may remove. A container or piece of paper with labels for each part is ideal to avoid confusion between the similiar-looking screws
7. Handle all parts with care. Place each piece you remove carefully down onto a stable surface
8. If a component does not come out easily, do not forcefully remove it.
9. Be careful when handling the motherboard.

10. Never attempt to remove the power source, a box attached to the side or bottom of the unit to which all cables are connected.
11. When removing any cables, wires or ribbons make sure to grasp the wire at the base or head to keep it from breaking
12. Be careful not to drop any small parts into unreachable areas such as into the computer fan or disk drive
13. Take note that the three most damaging things is a computer are moisture, shock and dust.

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

Tutorial -II

Programme: B.E
Course: Computer Organization

Term: Jan to May 2020
Course Code: CS45

Name: Keerthana Ningaraju	**Marks:** 10/10	**Date:** 28/01/2020
USN: 1M18CS059	**Signature of the Faculty:**	

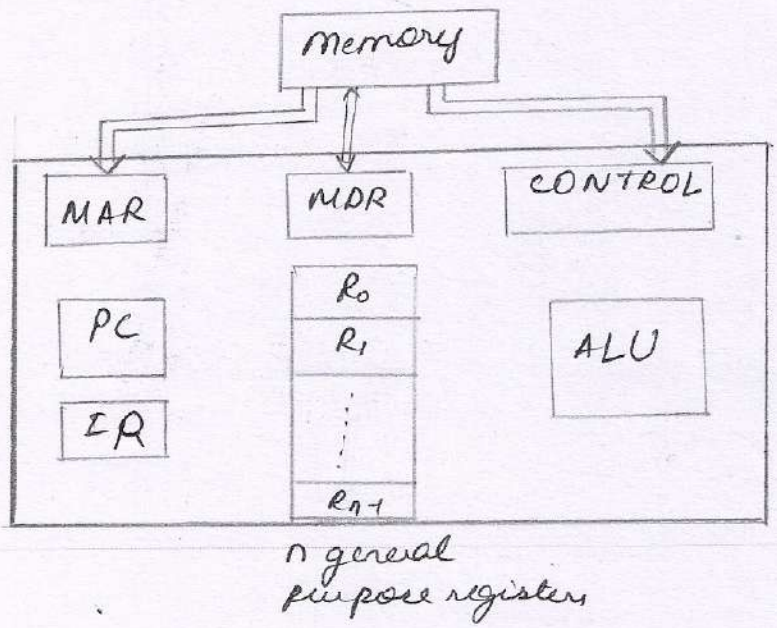
Activity II: Demonstrating Datapath and instruction execution stages using MarieSim Simulator

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.



n general purpose registers

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:

2. List out the steps required to execute an instruction

Soln. 6 steps: (i) Fetch instruction
(ii) Decode instruction
(iii) Perform ALU operation
(iv) Access memory
(v) Update register file
(vi) update program counter

3 (i)
```
LOAD G
ADD H
STORE A
LOAD I
ADD Y
STORE B
loop, LOAD A
      ADD F
      STORE F
      LOAB B
      SUBT one
      STORE B
      SKIPCOND 400
      JUMP loop
LOAD F
OUTPUT
HALT
G, DEC 7
H, DEC 3
Y, DEC 1
I, DEC 4
A, DEC 0
B, DEC 0
F, DEC 0
one, DEC 1
```

(ii) ~~LOAD~~

```
Loop, LOAD X
ADD B
STORE B
LOAD D
SUBT one
STORE D
SKIPCOND 400
JUMP Loop
Loop, LOAD Y
ADD C
STORE C
LOAD F
SUBT one
STORE F
SKIPCOND 400
JUMP Loop
Loop, LOAD Z
ADD C
STORE C
LOAD A
SUBT one
STORE A
SKIPCOND 400
JUMP Loop
LOAD B
SUBT C
STORE ans
LOAD ans
OUTPUT
HALT
A, DEC 10
B, DEC 8
C, DEC 2
F, DEC 4
one, DEC 1
D, DEC 8
X, DEC 0
Y, DEC 0
Z, DEC 0
```

RAMAIAH
Institute of Technology

Data Sheet

MARKS :

Name :	Keerthana Nirgaraju	Branch:	CSE
USN/Roll No. :	1MS18CS059	Sem/Sec:	IV 'B'
Subject :	CO Lab	Subject Code:	

4.(i) Processor clock : Processor circuits are controlled by a clock - it is a signal clock defines regular intervals - clock cycles to execute a machine instruction, the processor divides the action to be performed into a series of steps, each completed in 1 clock cycle. Length P of 1 clock cycle is important for processor performance, whose inverse is called the clock rate $R = 1/P$ cycles/sec is called Hertz, mega-millions, Giga-billions

(ii) Basic Performance Eqn : $T = \frac{N \times S}{R}$ where N is no of machine & language instruction

R = clock rate (in cycles/sec)

S = No of basic steps to execute 1 instruction

T = processor time

for high performance, reduce T (by reducing N and S, increasing R)

(iii) Pipelining and superscalar operation: Pipelining is the overlapping execution of several instructions. this reduces no of clock cycles, we can achieve a higher degree of concurrency. Parallel paths are created. Execution of several instructions in every clock cycle is called superscalar execution

(iv) clock rate: 2 possibilities to increase clock rate (improving IC tech, makes logic circuits faster, which reduce time to execute each step. P reduces, R is increased. secondly reducing amount of processing also works

(v) CISC and RISC

RISC Allows simple instructions which requires small number of basic steps to execute. A program will have more no of instructions i.e N = max, S = min.

CISC Instructions are complex, Number of basic steps to execute are more. A program will have less no of instructions i.e N = min, S = max

(vi) Compiler

translates high-level language into a sequence of machine instruction. Optimizes compiler reduces N x S, may rearrange program instructions to achieve better performance high-quality compiler must be closely linked to processor architecture, they are often designed at the same time, with much interaction to achieve best results

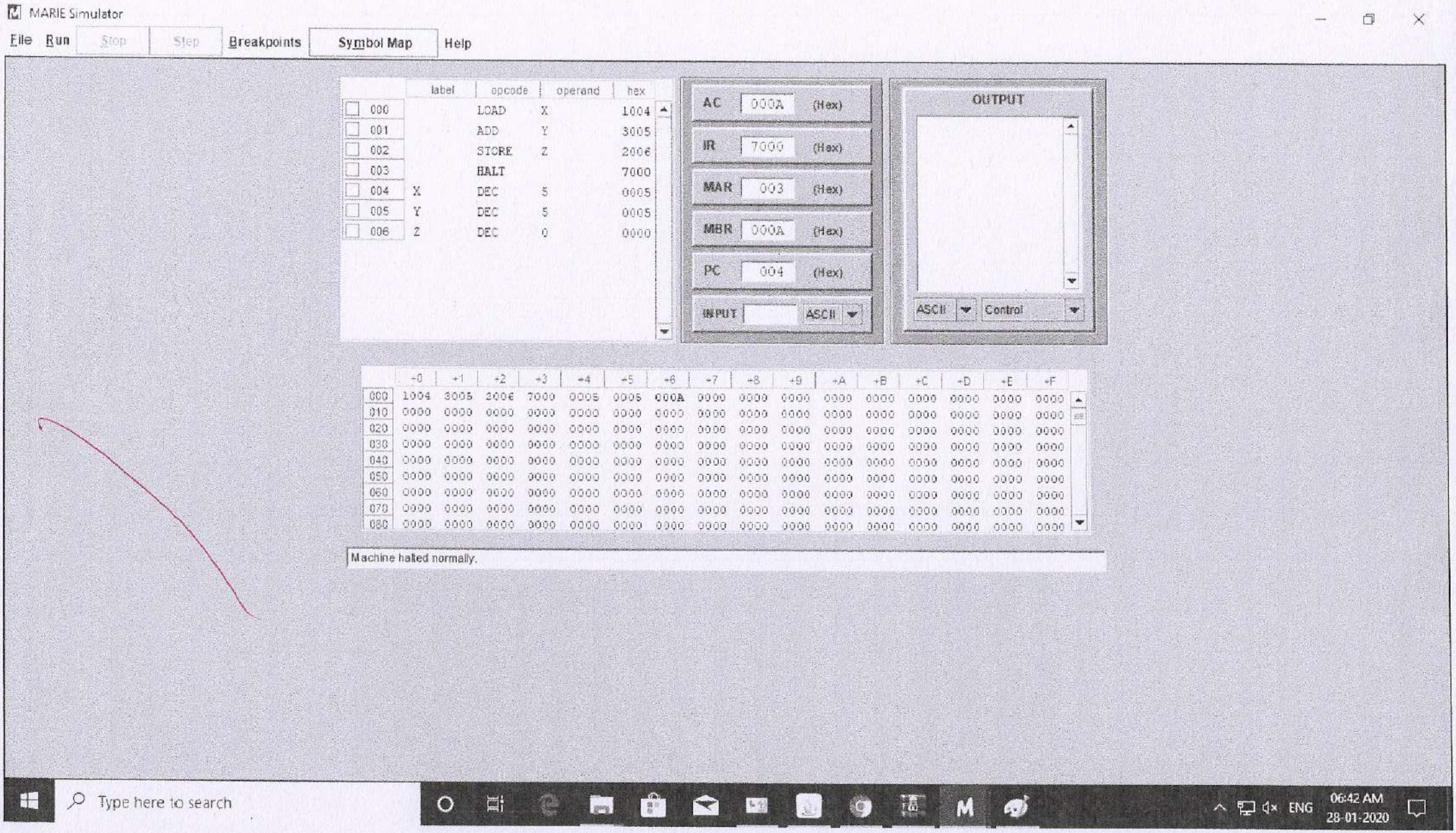
(vii) Performance measurements:

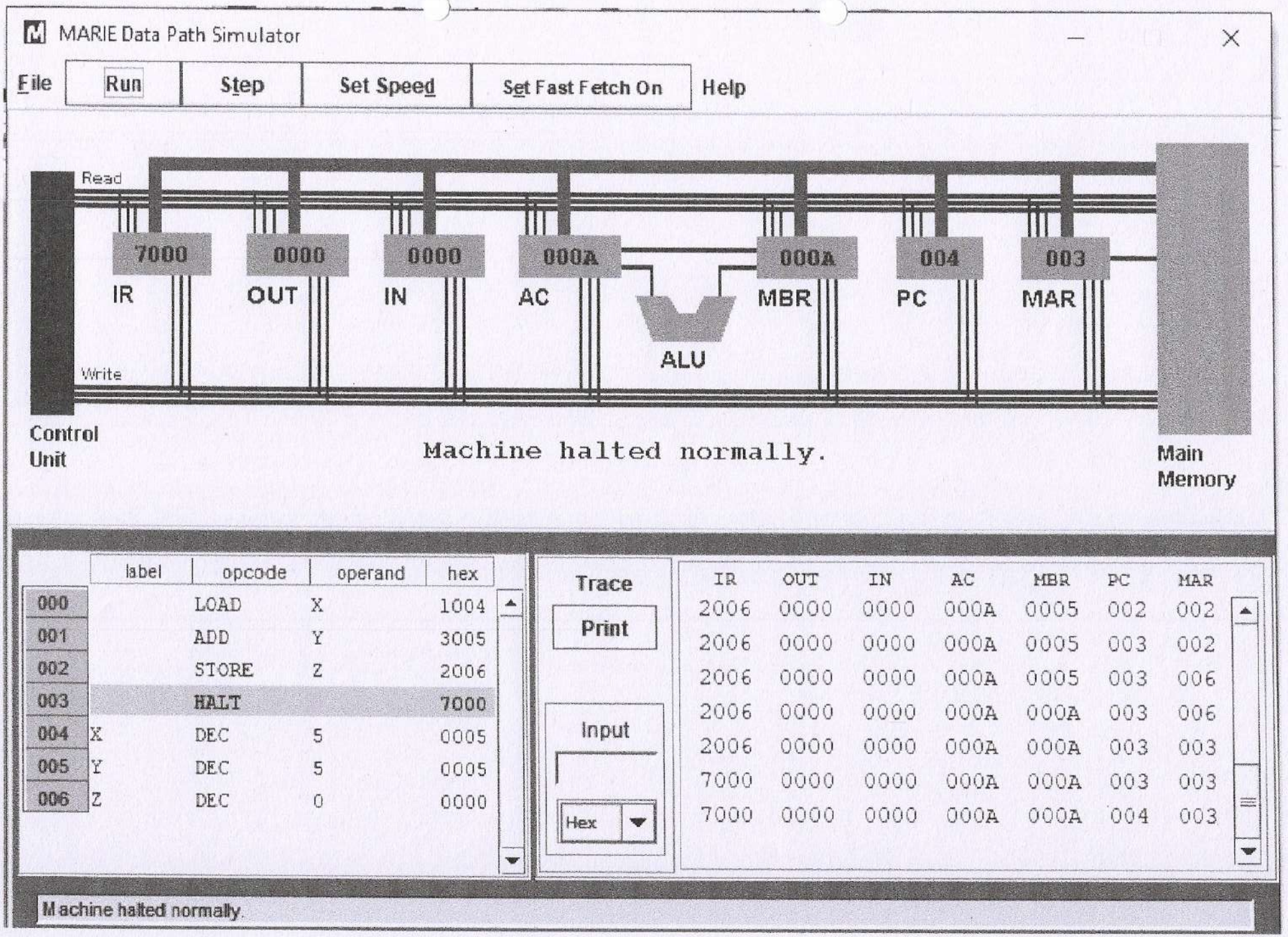
Used to evaluate effectiveness of new features. used in marketing process, to choose computer models. It is done using time factor taken to execute benchmarked programs SPEC4. ~~specific Per~~

specific Performance evaluation corporation

$$= \frac{\text{Running time on reference computer}}{\text{Running time on test computer}}$$

$$= \left(\prod_{i=1}^{n} SPEC_i \right)^{1/n}$$


MARIE Simulator
File Run Stop Step Breakpoints Symbol Map Help
label opcode operand hex
000 LOAD X 1004
001 ADD Y 3005
002 STORE Z 200E
003 HALT 7000
004 X DEC 5 0005
005 Y DEC 5 0005
006 Z DEC 0 0000
AC 000A (Hex)
IR 7000 (Hex)
MAR 003 (Hex)
MBR 000A (Hex)
PC 004 (Hex)
INPUT ASCII
OUTPUT
ASCII Control
+0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
000 1004 3005 200E 7000 0005 0005 000A 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
Machine halted normally.
Type here to search
ENG
06:42 AM
28-01-2020


MARIE Data Path Simulator
File
Run
Step
Set Speed
Set Fast Fetch On
Help
Read
Write
7000
IR
0000
OUT
0000
IN
000A
AC
ALU
000A
MBR
004
PC
003
MAR
Control Unit
Machine halted normally.
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 5 0005
006 Z DEC 0 0000
Trace
Print
Input
Hex
IR OUT IN AC MBR PC MAR
2006 0000 0000 000A 0005 002 002
2006 0000 0000 000A 0005 003 002
2006 0000 0000 000A 0005 003 006
2006 0000 0000 000A 000A 003 006
2006 0000 0000 000A 000A 003 003
7000 0000 0000 000A 000A 003 003
7000 0000 0000 000A 000A 004 003
Machine halted normally.

Assembly Listing for Ex2.mas

```
     Assembly listing for: Ex2.mas
              Assembled: Tue Jan 28 07:13:08 IST 2020

000 1011 |          LOAD G
001 3012 |          ADD H
002 2015 |          STORE A
003 1014 |          LOAD I
004 3013 |          ADD Y
005 2016 |          STORE B
006 1015 |  loop    LOAD A
007 3017 |          ADD F
008 2017 |          STORE F
009 1016 |          LOAD B
00A 4018 |          SUBT one
00B 2016 |          STORE B
00C 8400 |          SKIPCOND 400
00D 9006 |          JUMP loop
00E 1017 |          LOAD F
00F 6000 |          OUTPUT
010 7000 |          HALT
011 0007 |  G       DEC 7
012 0003 |  H       DEC 3
013 0001 |  Y       DEC 1
014 0004 |  I       DEC 4
015 0000 |  A       DEC 0
016 0000 |  B       DEC 0
017 0000 |  F       DEC 0
018 0001 |  one     DEC 1

Assembly successful.

          SYMBOL TABLE
          ------------------------------------------
```

Print

Close

Type here to search

ENG 07:15 AM 28-01-2020

Assembly Listing for Ex1.mas

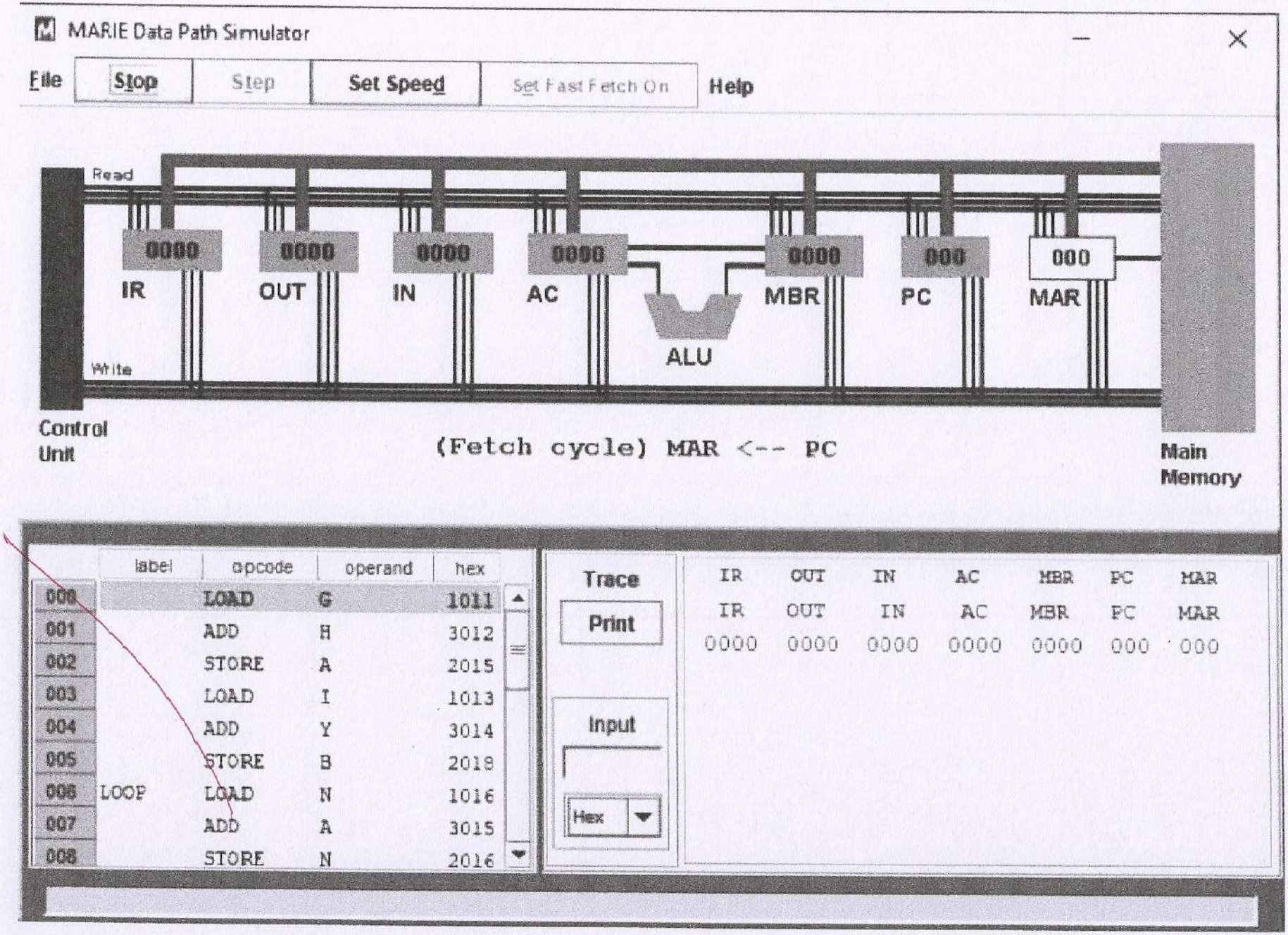
```
    Assembly listing for: Ex1.mas
              Assembled: Tue Jan 28 06:37:31 IST 2020

000 1004 |          LOAD X
001 3005 |          ADD Y
002 2006 |          STORE Z
003 7000 |          HALT
004 0005 |  X       DEC 5
005 0005 |  Y       DEC 5
006 0000 |  Z       DEC 0

Assembly successful.

        SYMBOL TABLE
        ------------------------------------------------
         Symbol | Defined | References
        --------+---------+-------------------------------
         X      |   004   | 000
         Y      |   005   | 001
         Z      |   006   | 002
        ------------------------------------------------
```

Print

Close

Type here to search

ENG 06:40 AM 28-01-2020


MARIE Data Path Simulator
File
Stop
Step
Set Speed
Set Fast Fetch On
Help
Read
Write
0000
IR
0000
OUT
0000
IN
0000
AC
ALU
0000
MBR
000
PC
000
MAR
Control Unit
(Fetch cycle) MAR <-- PC
Main Memory
label
opcode
operand
hex
000 LOAD G 1011
001 ADD H 3012
002 STORE A 2015
003 LOAD I 1013
004 ADD Y 3014
005 STORE B 2018
006 LOOP LOAD N 1016
007 ADD A 3015
008 STORE N 2016
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

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

Tutorial -III

Programme: B.E
Term: Jan to May 2020
Course: Computer Organization Course Code: CS45

Name: Keerthana Ningaraju	**Marks:** 10/10	**Date:** 04/02/2020
USN: 1MS18CS059	**Signature of the Faculty:**	11/2/2020

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:

Basic Arithmetic operations

(i)
```
MOV  R5, #10
MOV  R6, #20
ADD  R7, R6, R5
MUL  R8, R6, R5
SUB  R9, R6, R5
SWI  0X11
```

Load and store

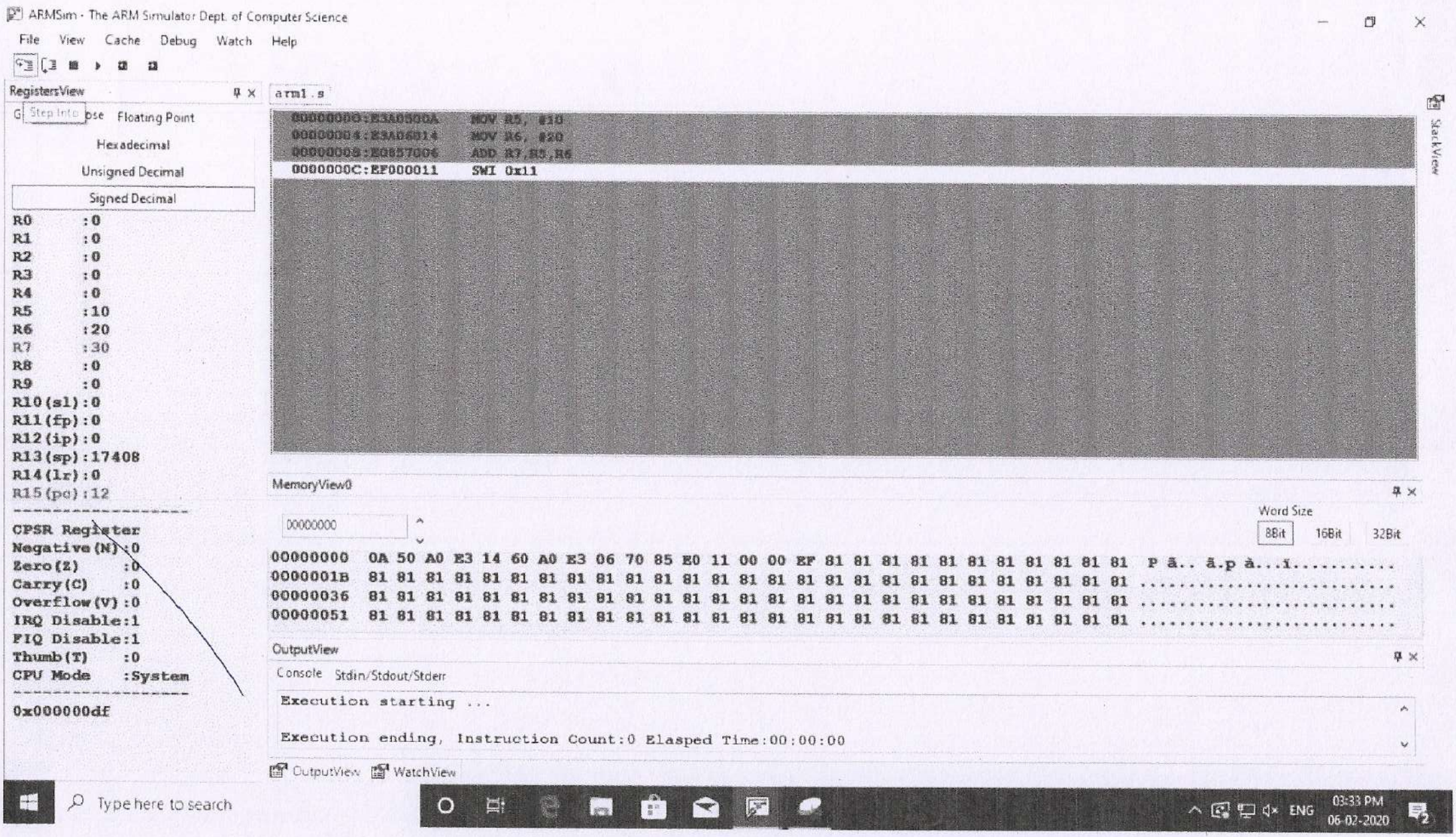
(ii)
```
MOV  R0, #10
MOV  R2, #0X00000040
STR  R0, [R2, #0]
LDR  R3, [R2, #0]
ADD  R4, R0, R3
SWI  0X11
```

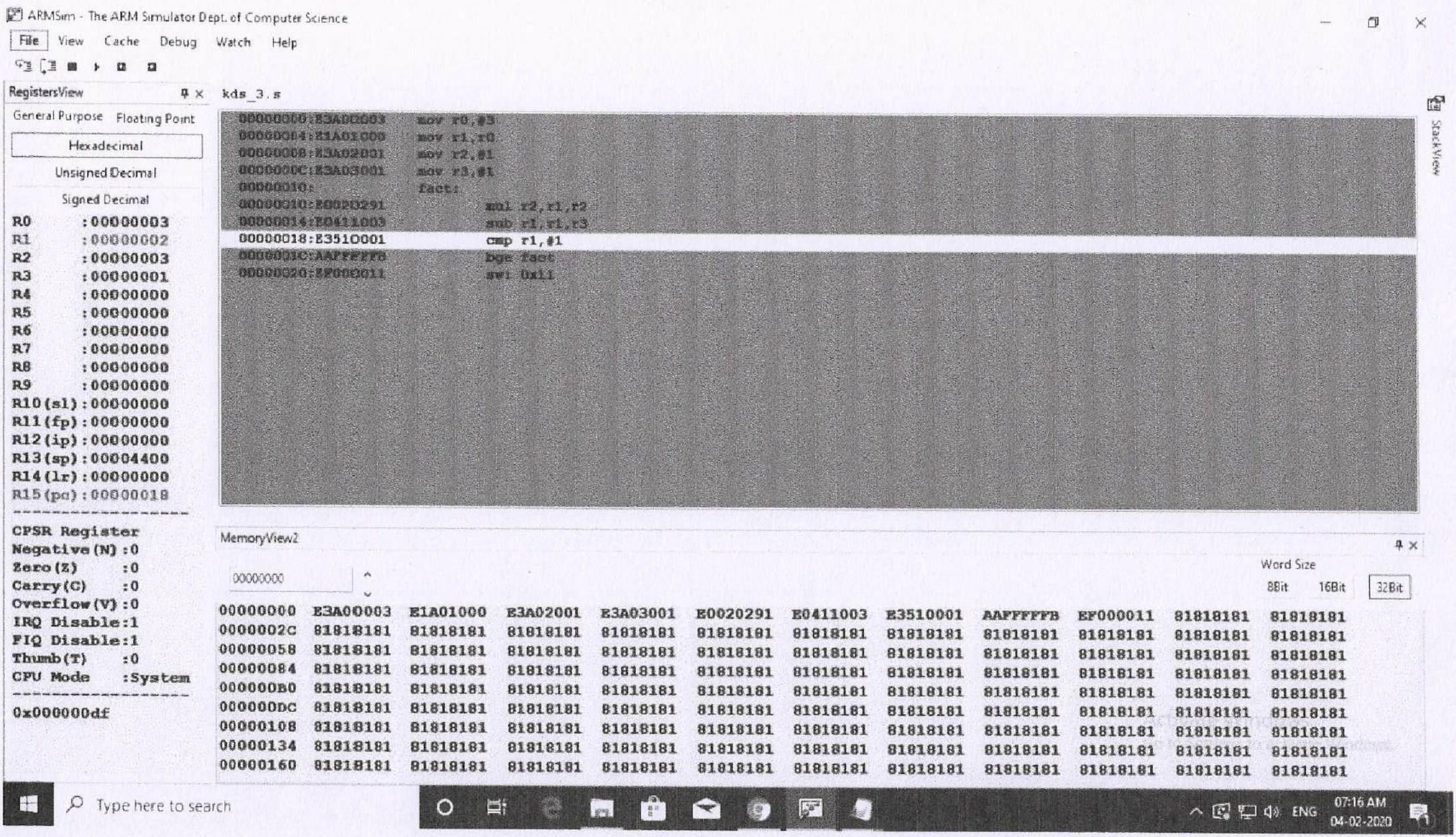
(iii) $f = (g+h) - (i+j)$

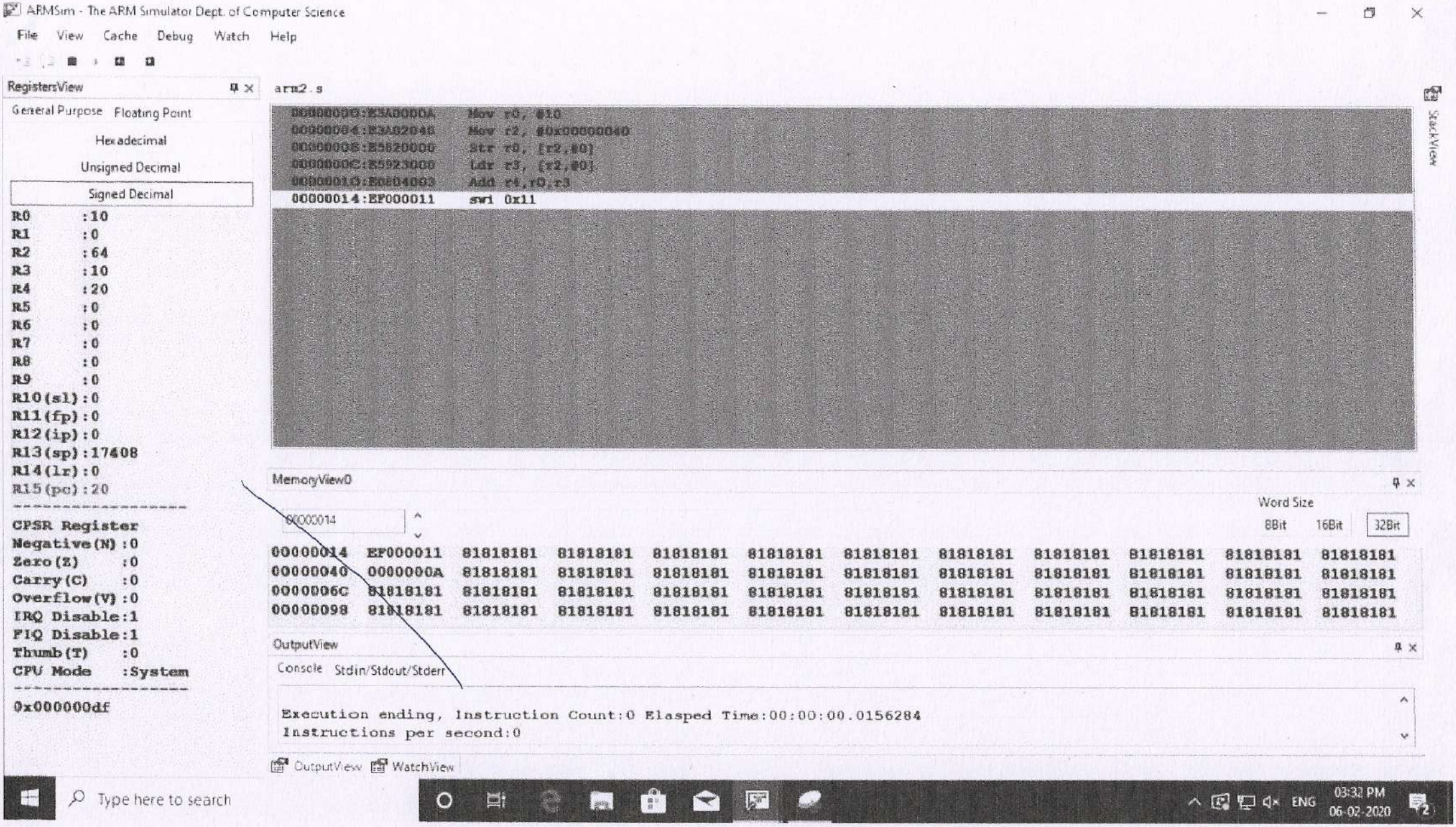
```
MOV  R1, #20
MOV  R2, #30
ADD  R3, R1, R2
MOV  R4, #40
MOV  R5, #50
ADD  R6, R1, R5
SUB  R7, R3, R6
SWI  0X11
```

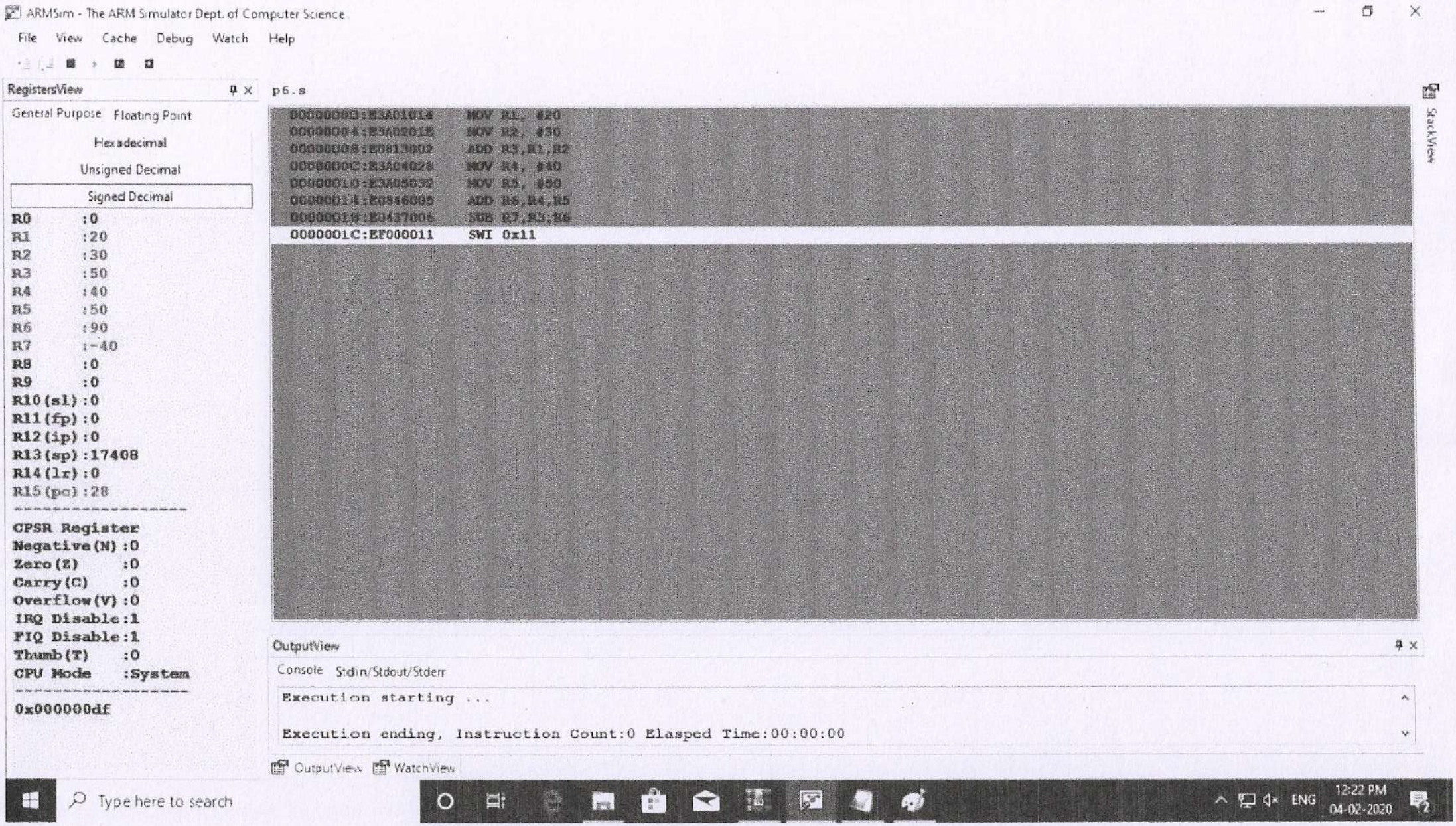
(iv) Factorial

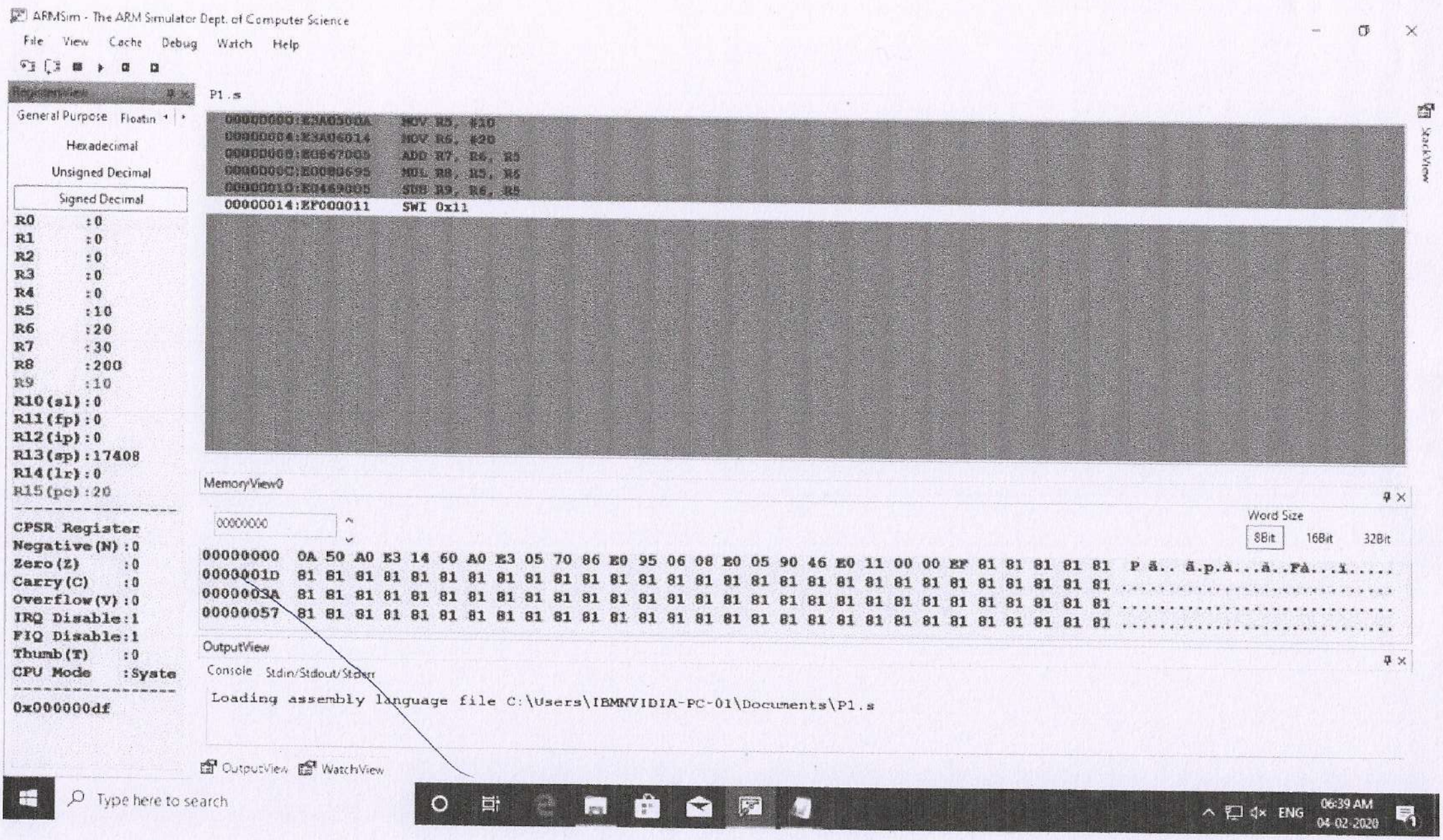
```
MOV  R0, #3
MOV  R1, R0
MOV  R2, #1
MOV  R3, #1
fact:
        MUL  R2, R1, R2
        SUB  R1, R1, R3
        CMP  R1, #1
        BGE  fact
        SWI  0X11
```


ARMSim - The ARM Simulator Dept. of Computer Science
File View Cache Debug Watch Help
RegistersView
Step Into
Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :0
R1 :0
R2 :0
R3 :0
R4 :0
R5 :10
R6 :20
R7 :30
R8 :0
R9 :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):17408
R14(lr):0
R15(pc):12
CPSR Register
Negative(N):0
Zero(Z) :0
Carry(C) :0
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T) :0
CPU Mode :System
0x000000df
arm1.s
00000000:E3A0500A MOV R5, #10
00000004:E3A06014 MOV R6, #20
00000008:E0857006 ADD R7,R5,R6
0000000C:EF000011 SWI 0x11
MemoryView0
00000000
Word Size
8Bit
16Bit
32Bit
00000000 0A 50 A0 E3 14 60 A0 E3 06 70 85 E0 11 00 00 EF 81 81 81 81 81 81 81 81 81 81 81 81
0000001B 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81
00000036 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81
00000051 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81
OutputView
Console Stdin/Stdout/Stderr
Execution starting ...
Execution ending, Instruction Count:0 Elasped Time:00:00:00
OutputView
WatchView
Type here to search
ENG
03:33 PM
06-02-2020


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 :00000002
R2 :00000003
R3 :00000001
R4 :00000000
R5 :00000000
R6 :00000000
R7 :00000000
R8 :00000000
R9 :00000000
R10(sl):00000000
R11(fp):00000000
R12(ip):00000000
R13(sp):00004400
R14(lr):00000000
R15(pc):00000018
CPSR Register
Negative(N):0
Zero(Z) :0
Carry(C) :0
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T) :0
CPU Mode :System
0x000000df
kds_3.s
00000000:E3A00003 mov r0,#3
00000004:E1A01000 mov r1,r0
00000008:E3A02001 mov r2,#1
0000000C:E3A03001 mov r3,#1
00000010: fact:
00000010:E0020291 mul r2,r1,r2
00000014:E0411003 sub r1,r1,r3
00000018:E3510001 cmp r1,#1
0000001C:AAFFFFFB bge fact
00000020:EF000011 swi 0x11
MemoryView2
00000000
Word Size
8Bit 16Bit 32Bit
00000000 E3A00003 E1A01000 E3A02001 E3A03001 E0020291 E0411003 E3510001 AAFFFFFB EF000011 81818181 81818181
0000002C 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
00000058 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
00000084 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
000000B0 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
000000DC 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
00000108 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
00000134 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
00000160 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
Type here to search
ENG
07:16 AM
04-02-2020


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 :10
R1 :0
R2 :64
R3 :10
R4 :20
R5 :0
R6 :0
R7 :0
R8 :0
R9 :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):17408
R14(lr):0
R15(pc):20
CPSR Register
Negative(N):0
Zero(Z) :0
Carry(C) :0
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T) :0
CPU Mode :System
0x000000df
arm2.s
00000000:E3A0000A Mov r0, #10
00000004:E3A02040 Mov r2, #0x00000040
00000008:E5820000 Str r0, [r2,#0]
0000000C:E5923000 Ldr r3, [r2,#0]
00000010:E0804003 Add r4,r0,r3
00000014:EF000011 swi 0x11
MemoryView0
00000014
Word Size
8Bit 16Bit 32Bit
00000014 EF000011 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
00000040 0000000A 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
0000006C 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
00000098 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
OutputView
Console Stdin/Stdout/Stderr
Execution ending, Instruction Count:0 Elasped Time:00:00:00.0156284
Instructions per second:0
OutputView WatchView
StackView
Type here to search
ENG
03:32 PM
06-02-2020


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 :0
R1 :20
R2 :30
R3 :50
R4 :40
R5 :50
R6 :90
R7 :-40
R8 :0
R9 :0
R10(sl) :0
R11(fp) :0
R12(ip) :0
R13(sp) :17408
R14(lr) :0
R15(pc) :28
CPSR Register
Negative(N) :0
Zero(Z) :0
Carry(C) :0
Overflow(V) :0
IRQ Disable :1
FIQ Disable :1
Thumb(T) :0
CPU Mode :System
0x000000df
p6.s
00000000:E3A01014 MOV R1, #20
00000004:E3A0201E MOV R2, #30
00000008:E0813002 ADD R3,R1,R2
0000000C:E3A04028 MOV R4, #40
00000010:E3A05032 MOV R5, #50
00000014:E0846005 ADD R6,R4,R5
00000018:E0437006 SUB R7,R3,R6
0000001C:EF000011 SWI 0x11
OutputView
Console Stdin/Stdout/Stderr
Execution starting ...
Execution ending, Instruction Count:0 Elasped Time:00:00:00
OutputView WatchView
QuickView
Type here to search
ENG
12:22 PM
04-02-2020


ARMSim - The ARM Simulator Dept. of Computer Science
File View Cache Debug Watch Help
P1.s
00000000:E3A0500A MOV R5, #10
00000004:E3A06014 MOV R6, #20
00000008:E0867005 ADD R7, R6, R5
0000000C:E0080695 MUL R8, R5, R6
00000010:E0469005 SUB R9, R6, R5
00000014:EF000011 SWI 0x11
General Purpose Floatin
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :0
R1 :0
R2 :0
R3 :0
R4 :0
R5 :10
R6 :20
R7 :30
R8 :200
R9 :10
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):17408
R14(lr):0
R15(pc):20
CPSR Register
Negative(N):0
Zero(Z) :0
Carry(C) :0
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T) :0
CPU Mode :Syste
0x000000df
MemoryView0
00000000
Word Size
8Bit 16Bit 32Bit
00000000 0A 50 A0 E3 14 60 A0 E3 05 70 86 E0 95 06 08 E0 05 90 46 E0 11 00 00 EF 81 81 81 81 81
0000001D 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81
0000003A 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81
00000057 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81
OutputView
Console Stdin/Stdout/Stderr
Loading assembly language file C:\Users\IBMNVIDIA-PC-01\Documents\P1.s
OutputView WatchView
StackView
Type here to search
ENG
06:39 AM
04-02-2020

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

Department of CSE

Programme: B.E **Term: Jan to May**
2019
Course: Computer Organization **Course Code:**
CS45

Activity IV: Executing ARM programs using ARMsim simulator.

Name: Keerthana Ningaraju	**Marks:** /10	**Date:** 11/02/2020
USN: 1MS18CS059	**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.

Results/Conclusions and Snapshots: Take the snap shot of registers file and memory view

RAMAIAH
Institute of Technology

Data Sheet

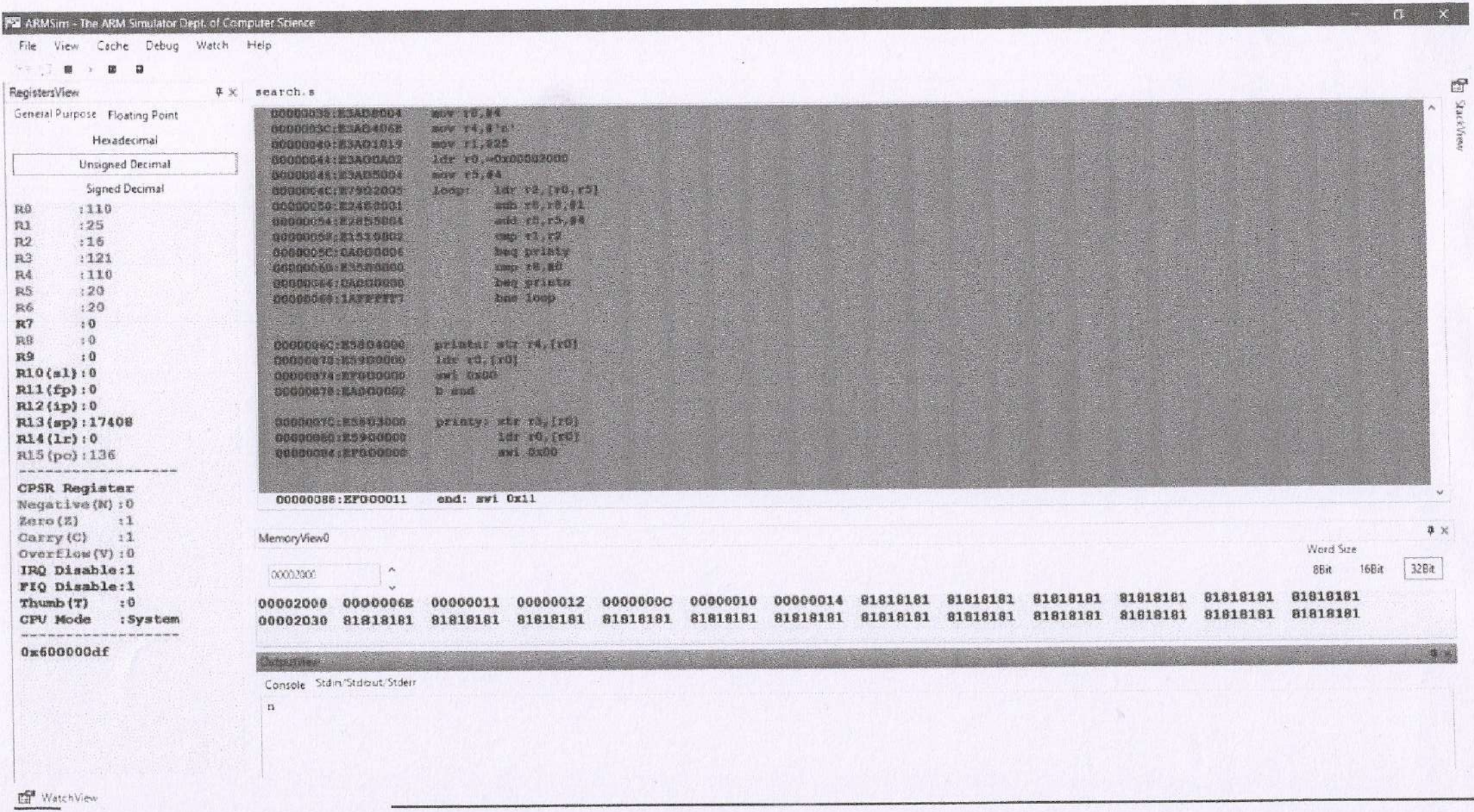
MARKS :

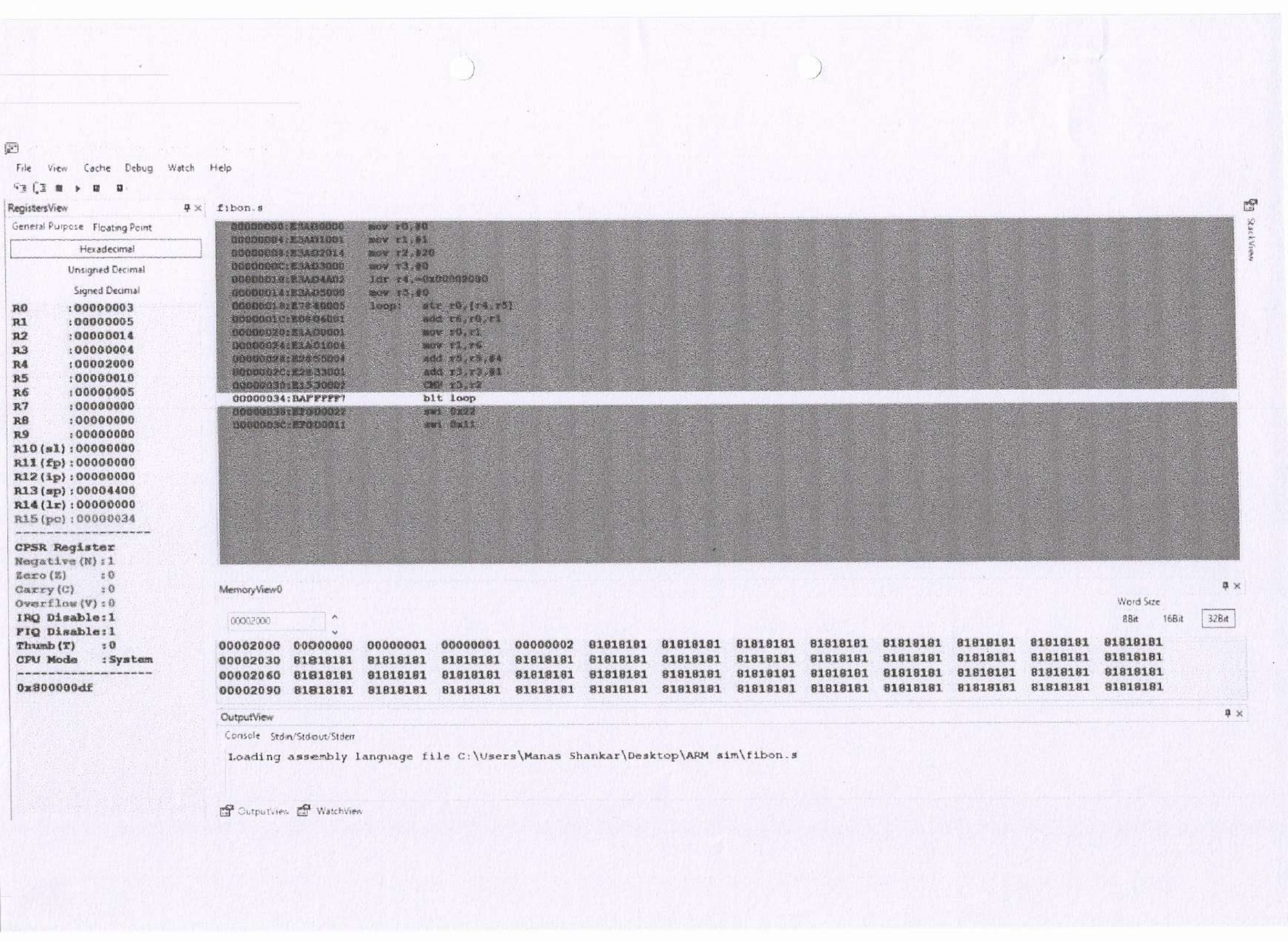
Name :	Kushara Nagaraju	Branch:	CSE
USN/Roll No. :	1MS18CS059	Sem/Sec:	IV 'B'
Subject :	CO Lab	Subject Code:	

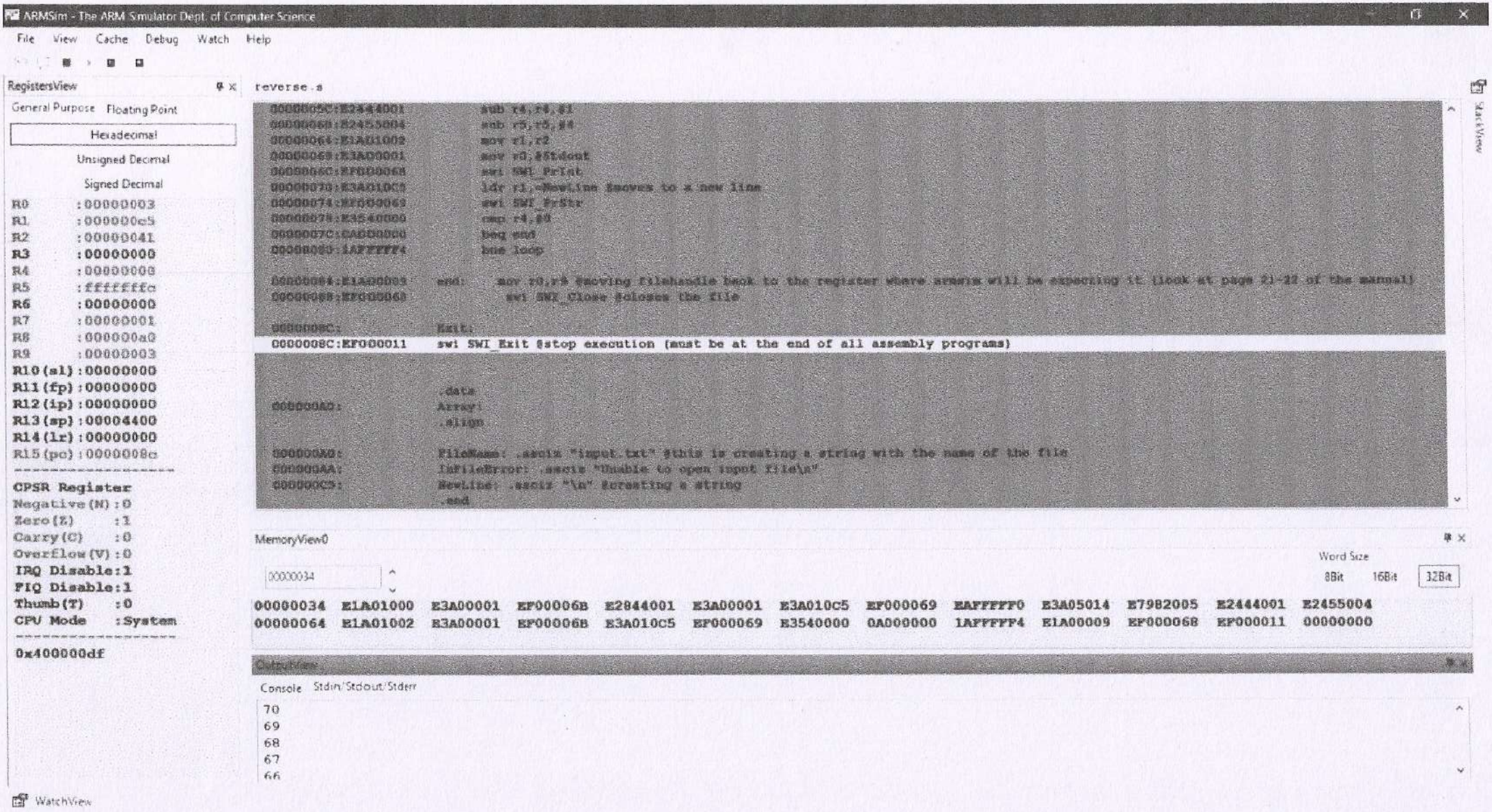
1).
```
mov   R0, #0
mov   R1, #1
mov   R2, #5
mov   R3, #0
LDR   R4, =0x00002000
mov   R6, #0
LOOP: STR   R0, [R4, R5]
      ADD   R6, R0, R1
      mov   R0, R1
      mov   R1, R6
      ADD   R5, R5, #4
      ADD   R3, R3, #1
      CMP   R3, R2
      BLT   LOOP

      SWI 0x11
```

2).
```
mov   R0, #4
mov   R4, #'n'
mov   R1, #25
LDR   R0, =0x00002000
mov   R5, #4
LOOP: LDR   R2, [R0, R5]
      SUB   R0, R0, #1
      ADD   R5, R5, #4
      CMP   R1, R2
      BEQ   print
```

```
        ldr  r0, = Array
        bes    afterloop
        ldr  r0;[r8, r5]
        add  r8, r5, #4
        mov  r1, r0
        mov  r0, # ~~start~~ stdout
        swi   SWI_PrInt
        add    r4, r4, #1
        ldr     r1, = Newline
       swi   SWI_Prstr
       bel  loopstart

afterloop:
  mov  r5, #20
  loop: ldr  r2, [r8, r5]
        sub  r4, r4, #1
        sub  r5, r5, #4
        mov  r1, r2
        mov  r0, ~~#start~~ stdout
        swi   SWI_PrInt
        ldr   r1, = Newline
        swi   SWI_Prstr
        cmp   r4, #0
        beq    end
        bne    loop

end:  mov r0, r4
      swi 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 :110
R1 :25
R2 :16
R3 :121
R4 :110
R5 :20
R6 :20
R7 :0
R8 :0
R9 :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):17408
R14(lr):0
R15(pc):136
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
search.s
00000038:E3A06004 mov r0,#4
0000003C:E3A0406E mov r4,#'n'
00000040:E3A01019 mov r1,#25
00000044:E3A00A02 ldr r0,=0x00002000
00000048:E3A05004 mov r5,#4
0000004C:E7902005 loop: ldr r2,[r0,r5]
00000050:E2455001 sub r5,r5,#1
00000054:E2055004 and r5,r5,#4
00000058:E1510002 cmp r1,r2
0000005C:0A000006 beq print
00000060:E3550000 cmp r5,#0
00000064:0A000000 beq print
00000068:1AFFFFF7 bne loop
0000006C:E5804000 print: str r4,[r0]
00000070:E5900000 ldr r0,[r0]
00000074:EF000000 swi 0x00
00000078:EA000002 b end
0000007C:E5803000 print: str r3,[r0]
00000080:E5900000 ldr r0,[r0]
00000084:EF000000 swi 0x00
00000388:EF000011 end: swi 0x11
MemoryView0
00002000
Word Size
8Bit 16Bit 32Bit
00002000 0000006E 00000011 00000012 0000000C 00000010 00000014 81818181 81818181 81818181 81818181 81818181 81818181
00002030 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
Console Stdin/Stdout/Stderr
n
WatchView


File View Cache Debug Watch Help
RegistersView
General Purpose Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :00000003
R1 :00000005
R2 :00000014
R3 :00000004
R4 :00002000
R5 :00000010
R6 :00000005
R7 :00000000
R8 :00000000
R9 :00000000
R10(sl):00000000
R11(fp):00000000
R12(ip):00000000
R13(sp):00004400
R14(lr):00000000
R15(pc):00000034
CPSR Register
Negative(N):1
Zero(Z) :0
Carry(C) :0
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T) :0
CPU Mode :System
0x800000df
fibon.s
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:E7840005 loop: str r0,[r4,r5]
0000001C:E0806001 add r6,r0,r1
00000020:E1A00001 mov r0,r1
00000024:E1A01006 mov r1,r6
00000028:E2855004 add r5,r5,#4
0000002C:E2833001 add r3,r3,#1
00000030:E1530002 CMP r3,r2
00000034:BAFFFFF7 blt loop
00000038:EF000022 swi 0x22
0000003C:EF000011 swi 0x11
MemoryView0
00002000
Word Size
8Bit 16Bit 32Bit
00002000 00000000 00000001 00000001 00000002 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101
00002030 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101
00002060 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101
00002090 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101
OutputView
Console Stdin/Stdout/Stderr
Loading assembly language file C:\Users\Manas Shankar\Desktop\ARM sim\fibon.s
OutputView WatchView


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 :000000c5
R2 :00000041
R3 :00000000
R4 :00000000
R5 :fffffffc
R6 :00000000
R7 :00000001
R8 :000000a0
R9 :00000003
R10(sl) :00000000
R11(fp) :00000000
R12(ip) :00000000
R13(sp) :00004400
R14(lr) :00000000
R15(pc) :0000008c
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
WatchView
reverse.s
00000058:E2444001 sub r4,r4,#1
0000005C:E2455004 sub r5,r5,#4
00000060:E1A01002 mov r1,r2
00000064:E3A00001 mov r0,#1 @stdout
00000068:EF000068 swi SWI_Print
0000006C:E3A010C5 ldr r1,=NewLine @moves to a new line
00000070:EF000069 swi SWI_PrStr
00000074:E3540000 cmp r4,#0
00000078:0A000000 beq end
0000007C:1AFFFFF4 bne loop
00000080:E1A00009 end: mov r0,r9 @moving filehandle back to the register where armsim will be expecting it (look at page 2)-22 of the manual)
00000084:EF000068 swi SWI_Close @closes the file
00000088: Exit:
0000008C:EF000011 swi SWI_Exit @stop execution (must be at the end of all assembly programs)
.data
000000A0: Array:
.align
000000A0: FileName: .asciz "input.txt" @this is creating a string with the name of the file
000000AA: InFileError: .asciz "Unable to open input file\n"
000000C9: NewLine: .asciz "\n" @creating a string
.end
MemoryView0
00000034
Word Size
8Bit 16Bit 32Bit
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

Computer Organization Lab

Activity VI: Designing memory system using Logisim simulator.

Name:Keerthana Ningaraju	**Date: 20/04/2020**
USN:1MS18CS059	**Signature of the Faculty:**

Objective: To simulate the writing operation on memory.

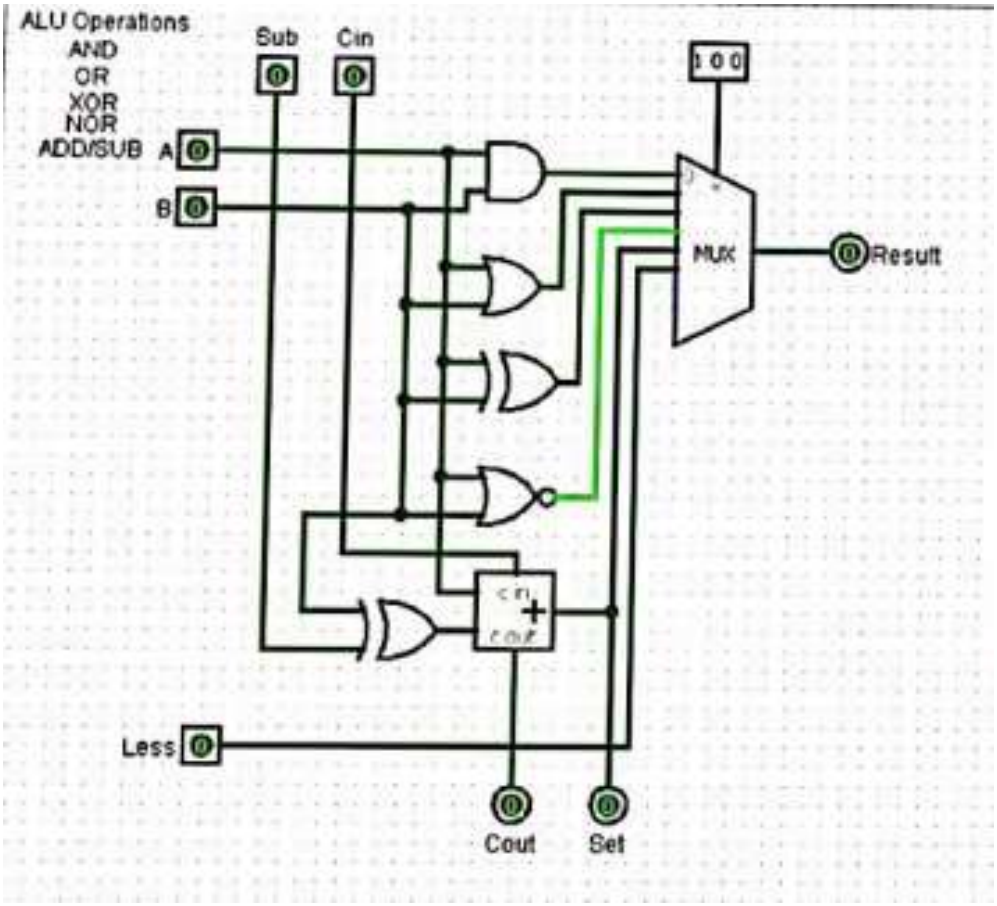
Simulator Description: Logisim is an educational tool for designing and simulating digital logic circuits. With its simple toolbar interface and simulation of circuits as you build them, it is simple enough to facilitate learning the most basic concepts related to logic circuits. With the capacity to build larger circuits from smaller sub circuits, and to draw bundles of wires with a single mouse drag, Logisim can be used (and is used) to design and simulate entire CPUs for educational purposes.

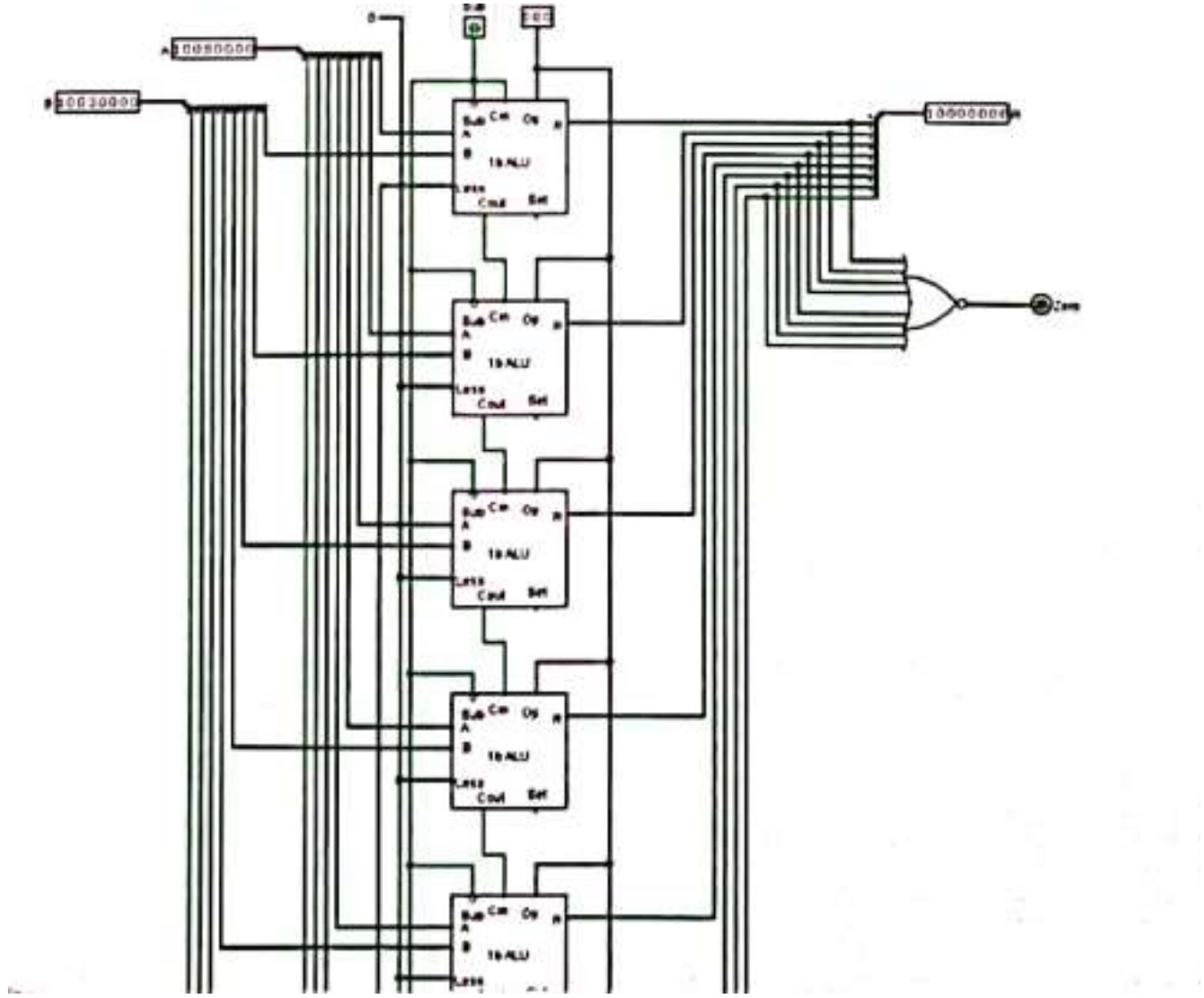
Activity to be performed by students

List out the steps in designing memory system

1. Add the two i/ps pins, name them A and B
2. Add an, and ex-or, nor gates and a 1-bit adder
3. Connect the A's and B's of all the gates to their respective pins
4. Add an output pin and name is Result
5. Add a 6-bit multiplier with 3 select bits
6. Connect outputs of all the gates to the mux.
7. Connect 3 bit input to mux
8. Add i/p pin to Cin, and output pin to cout.
9. Add an ex-or gate, connect its o/p to cout, the first i/p must to be continued B and the second to another i/p pin sub
10. Add another i/p and name it less, Connect it to the mux.
11. Add an o/p pin and name it set, connect it to the o/p of the adder unit.

Observations and Snapshots:





MARKS FOR DIFFERENT CRITERIA	MARKS OBTAINED
Exploring of Tool : 2M	
Execution: 6M	
Documenting the Work and Results:2M	
Total: 10M	

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

Department of CSE

Programme: B.E **Term: Jan to May 2019**
Course: Computer Organization **Course Code: CS45**

Activity VI: Designing memory system using Logisim simulator.

Name: Keerthana Ningaraju	**Marks: /10**	**Date: 20/04/2020**
USN: 1MS18CS059	**Signature of the Faculty:**	

Objective: To simulate the writing operation on memory.

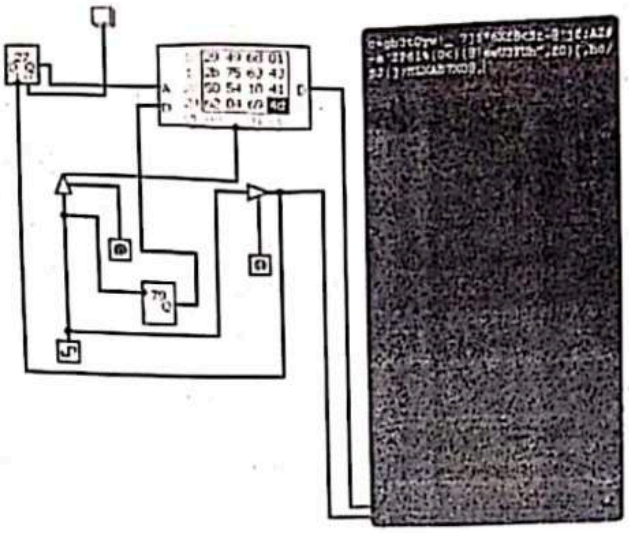
Simulator Description: Logisim is an educational tool for designing and simulating digital logic circuits. With its simple toolbar interface and simulation of circuits as you build them, it is simple enough to facilitate learning the most basic concepts related to logic circuits. With the capacity to build larger circuits from smaller sub circuits, and to draw bundles of wires with a single mouse drag, Logisim can be used (and is used) to design and simulate entire CPUs for educational purposes.

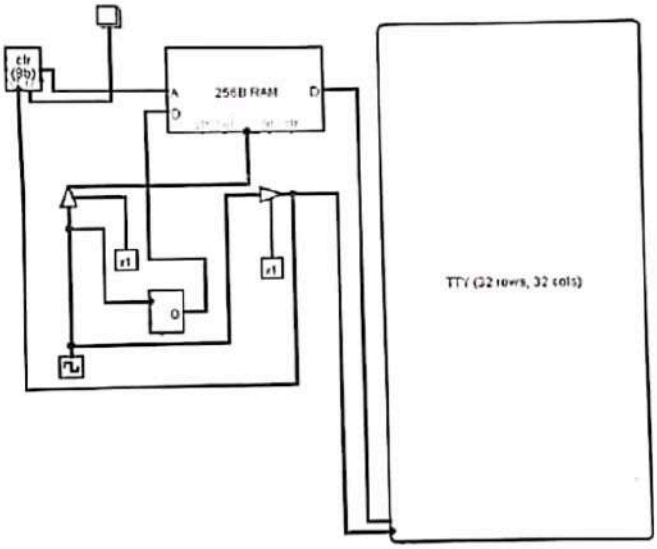
Activity to be performed by students:

List out the steps in designing memory system

1) Add a RAM with seperate load & store selected
2) Add a counter and count 0 to A of the RAM
3) Add a controller buffer and connect its output to the paper
4) Add a controller buffer and connect its output to the RAM
5) Add a TTY unit & with 32 rows and columns, make the connections with RAM
6) Add a 7-bit random number generator, connect a end to D
7) Add another controlled buffer, connect the TTY, also add an input pin to the buffer
8) Connect the output of the second buffer to the counter
9) connect a button to the counter

Observations and Snapshots:





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

Department of CSE

Programme: B.E **Term: Jan to May 2019**
Course: Computer Organization **Course Code: CS45**

Activity VII: To simulate advantages of using pipeline technique in executing a program.

Name: Keerthana Ningaraju	**Marks: /10**	**Date: 20/05/2020**
USN: 1MS18CS059	**Signature of the Faculty:**	

Objective: To learn and analyze the performance of the CPU by overlapping of instructions using CPUOS-SIM simulator.

Simulator Used: CPUOS-SIM is a software development environment for the simulation of simple computers. It was developed by Dale Skrien to help users to understand computer architectures.

Modern CPU's contain several semi-independent circuits involved in decoding and executing each machine instruction. Separate circuit elements perform each of these typical steps:

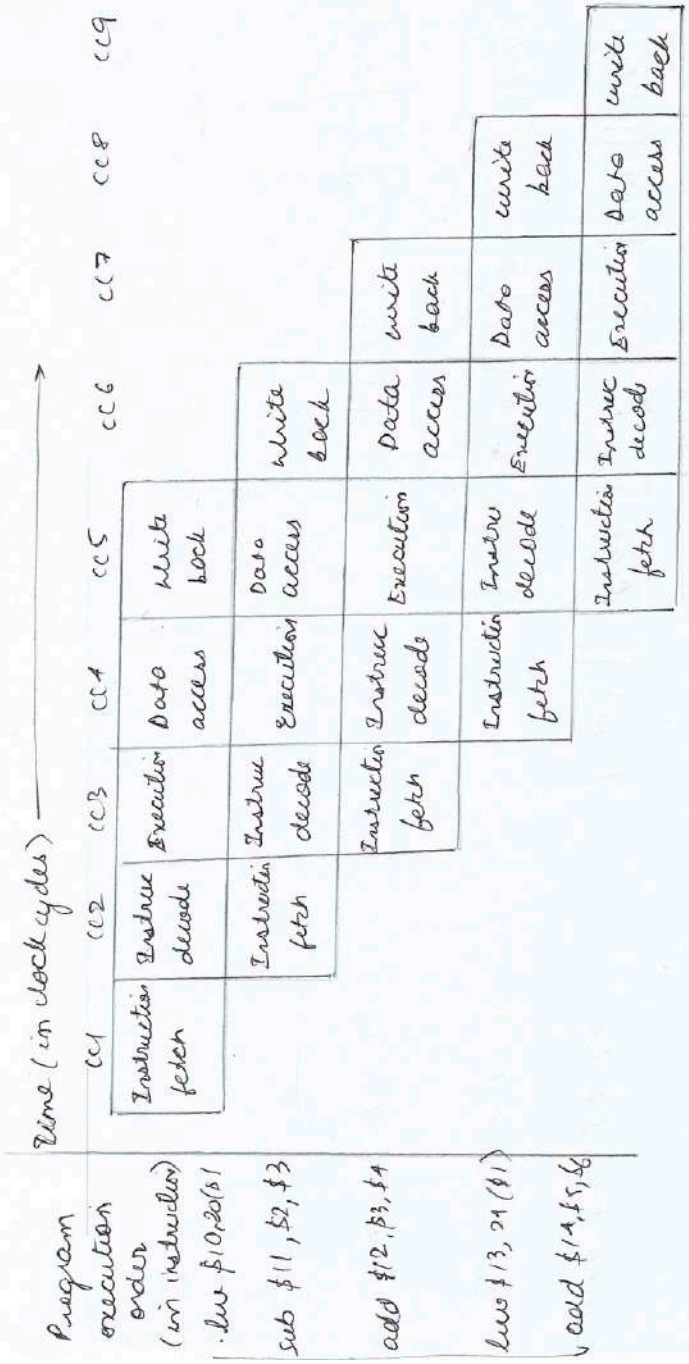
- Fetch the next instruction from memory into an internal CPU register.
- Decode the instruction to determine which function sub-circuits it requires.
- Read any input operands required from high-speed registers or directly from memory.
- Execute the operation using the selected sub-circuits.
- Write any output results to high-speed registers or directly to memory.

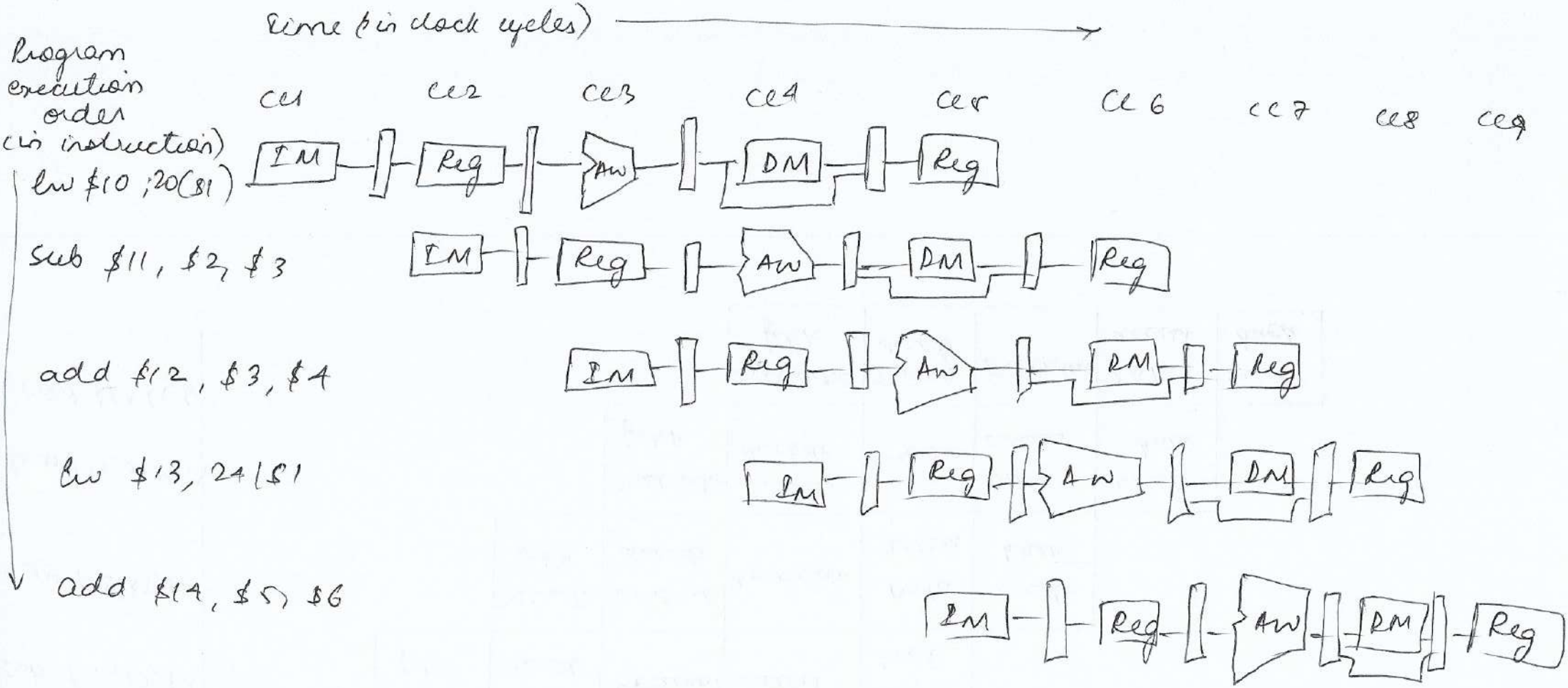
Separate sections of the CPU circuitry are used for each of these steps. This allows these circuit sections to be arranged into a sequential pipeline, with the output of one step feeding into the next step.

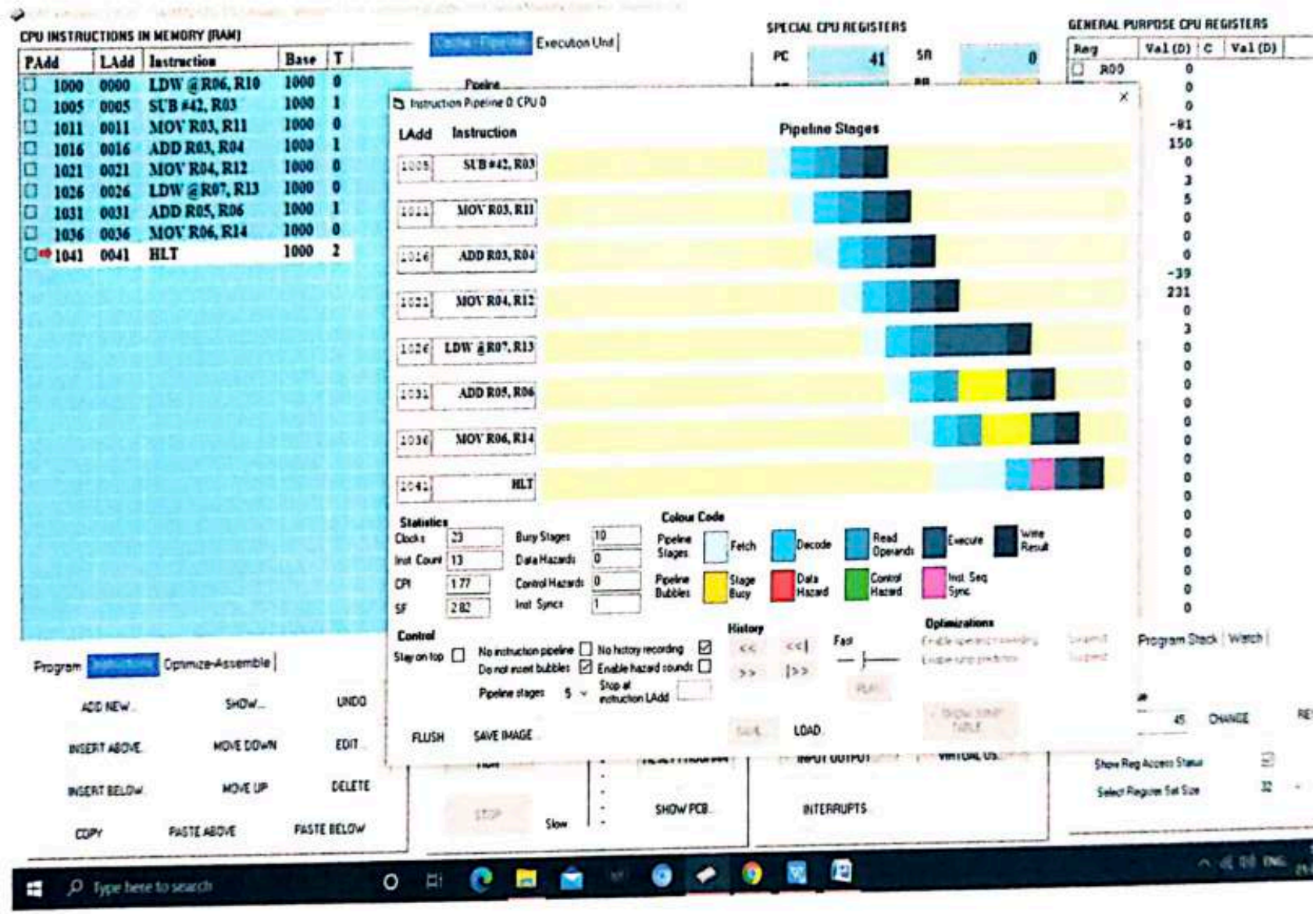
Activity to be performed by students:

With diagram demonstrate the execution of the following instructions using pipelining technique.

lw $10,20($1)
sub $11, 42, $3
add $12, $3, $4
lw $13, 24($1)
add $14, $5, $6




Time (in clock cycles)
Program execution order (in instruction)
cc1
cc2
cc3
cc4
cc5
cc 6
cc 7
cc 8
cc 9
lw $10, 20($1)
Sub $11, $2, $3
add $12, $3, $4
lw $13, 24($1)
add $14, $5, $6
IM
Reg
AW
DM
Reg


CPU INSTRUCTIONS IN MEMORY (RAM)
PAdd LAdd Instruction Base T
1000 0000 LDW @R06, R10 1000 0
1005 0005 SUB #42, R03 1000 1
1011 0011 MOV R03, R11 1000 0
1016 0016 ADD R03, R04 1000 1
1021 0021 MOV R04, R12 1000 0
1026 0026 LDW @R07, R13 1000 0
1031 0031 ADD R05, R06 1000 1
1036 0036 MOV R06, R14 1000 0
1041 0041 HLT 1000 2
Program
Optimize-Assemble
ADD NEW...
SHOW...
UNDO
INSERT ABOVE
MOVE DOWN
EDIT
INSERT BELOW
MOVE UP
DELETE
COPY
PASTE ABOVE
PASTE BELOW
Execution Unit
Pipeline
SPECIAL CPU REGISTERS
PC 41
SR 0
GENERAL PURPOSE CPU REGISTERS
Reg Val (D) C Val (D)
R00 0
0
0
-81
150
0
3
5
0
0
0
-39
231
0
3
0
0
0
0
0
0
0
0
0
0
0
0
0
0
Instruction Pipeline 0 CPU 0
LAdd Instruction
Pipeline Stages
1005 SUB #42, R03
1011 MOV R03, R11
1016 ADD R03, R04
1021 MOV R04, R12
1026 LDW @R07, R13
1031 ADD R05, R06
1036 MOV R06, R14
1041 HLT
Statistics
Clocks 23
Inst Count 13
CPI 1.77
SF 2.82
Busy Stages 10
Data Hazards 0
Control Hazards 0
Inst Syncs 1
Colour Code
Pipeline Stages
Fetch
Decode
Read Operands
Execute
Write Result
Pipeline Bubbles
Stage Busy
Data Hazard
Control Hazard
Inst. Seq. Sync
Control
Stay on top
No instruction pipeline
Do not insert bubbles
No history recording
Enable hazard sounds
Pipeline stages 5
Stop at instruction LAdd
History
Fast
Optimizations
FLUSH
SAVE IMAGE
LOAD
Program Stack
Watch
CHANGE
Show Reg Access Status
Select Register Set Size
32
STOP
Slow
SHOW PCB
INTERRUPTS
Type here to search

Observations and Snapshots: Take the snap shot of CPU statistics and pipeline design.