Microprocessor and Computer Architecture Laboratory
UE19CS256
4th Semester, Academic Year 2020-21

Working on Pipeline and Cache simulator

Name: **RAMYA NARASIMHA PRABHU**	SRN: **PES1UG19CS380**	Section: **F2**

Week: **9** Date: **9/4/2021**

Lab 9

Task1: 5 stage pipeline Simulator

Execute these instructions using 5 stage pipeline - MIPS architecture simulator.

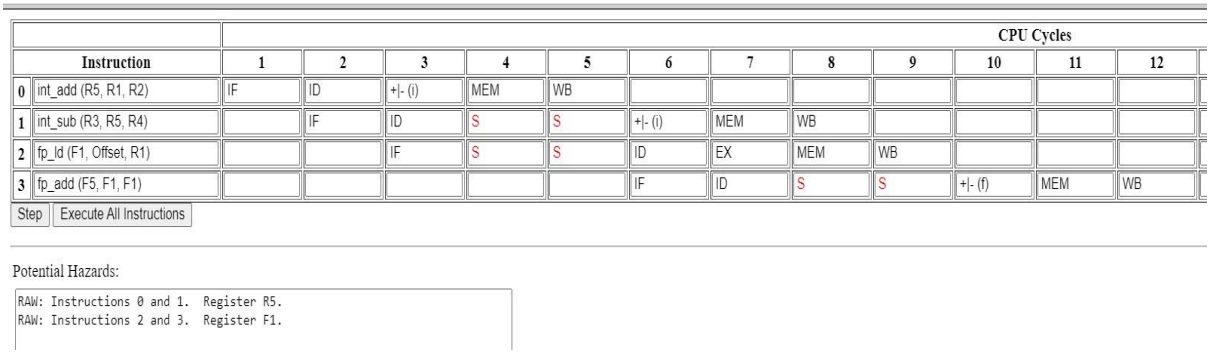
ADD R0, R1, R2

SUB R3, R0, R4

FP_LOAD F1, Offset,R1

FP_ADD F5,F1,F1

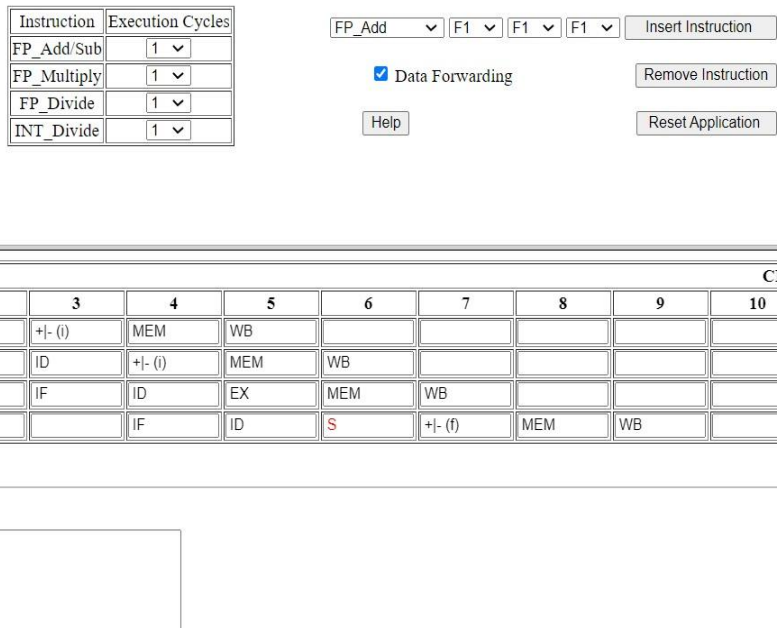
Without data forwarding:



Potential Hazards:

```
RAW: Instructions 0 and 1.  Register R5.
RAW: Instructions 2 and 3.  Register F1.
```

With data forwarding:



Instruction	Execution Cycles
FP_Add/Sub	1
FP_Multiply	1
FP_Divide	1
INT_Divide	1

FP_Add | F1 | F1 | F1 | Insert Instruction

Data Forwarding | Remove Instruction

Help | Reset Application

	Instruction	1	2	3	4	5	6	7	8	9	10
0	int_add (R5, R1, R2)	IF	ID	+|- (i)	MEM	WB					
1	int_sub (R3, R5, R4)		IF	ID	+|- (i)	MEM	WB				
2	fp_ld (F1, Offset, R1)			IF	ID	EX	MEM	WB			
3	fp_add (F5, F1, F1)				IF	ID	S	+|- (f)	MEM	WB	

Step | Execute All Instructions

Potential Hazards:

RAW: Instructions 2 and 3. Register F1.

Observe the following and note down the results.

1. *Is there a data dependency among the instructions?*

Ans. Yes.

In the case without data forwarding, there is a RAW data hazard between instructions 0 and 1 on R5 and between instructions 2 and 3 on F1.

With data forwarding - Instructions 2 and 3, Register F1
It occurs when the value produced by the first instruction is required by the subsequent instruction.

2. *How many stall states have been introduced?*

Ans. 6 stall states are introduced (without data forwarding).

3. *If data forwarding is applied how many stall states have been reduced?*

Ans. 5 stall states are reduced (leaving only 1).

4. *Mention the total number of clock cycles used with and without data forwarding.*

Ans.

Without data forwarding: 12 cycles

With data forwarding: 9 cycles

Task2: Cache Simulator

1. A computer system uses 16-bit memory addresses. It has a 2K-byte cache organized in a direct-mapped manner with 64 bytes per cache block. Assume that the size of each memory word is 1 byte.

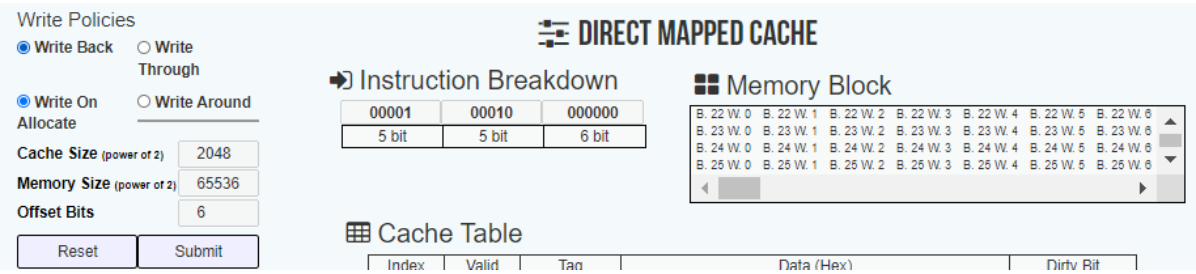
a) Calculate the number of bits in each of the Tag, Block, and Word fields of the memory address using direct mapped Cache.

Ans.

Total no. of address bits = 16
Block size = 64 bytes = 2^6 bytes = 2^6 words

No. of bits in the Word field = 6
Cache size = 2K-byte = 2x1024 = 2^{11} bytes
Number of cache blocks = Cache size / Block size = $2^{11}/2^6 = 2^5$
No. of bits in the Block field = 5

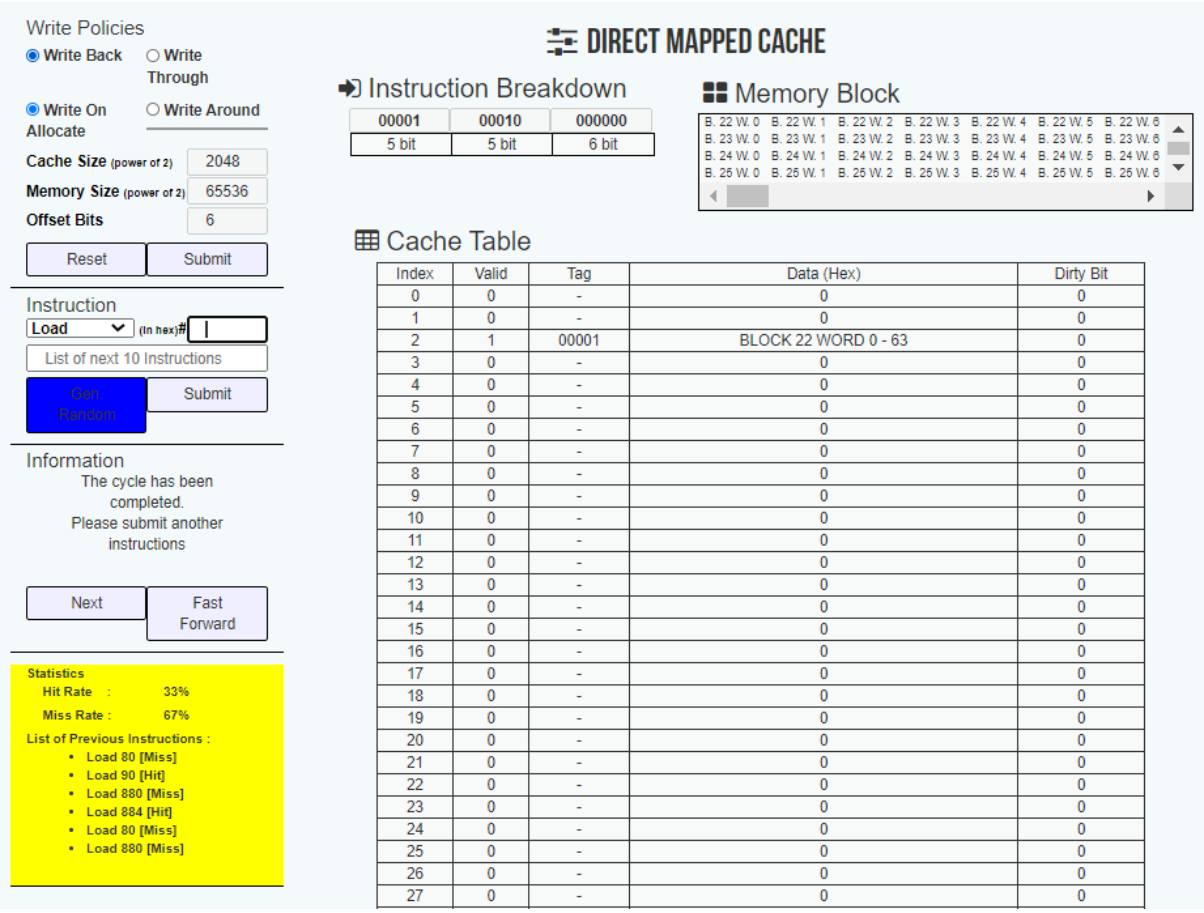
No. of bits in the Tag field = 16 - 6 - 5 = 5
Therefore, **No. of bits in Tag=5, Block=5, Word=6.**



b) When a program is executed, the processor reads data sequentially from the following word addresses: 128, 144, 2176, 2180, 128, 2176. All the above addresses are shown in decimal values (Convert the address to hexadecimal equivalent of the decimals given above and submit in the simulator). Assume that the cache is initially empty. For each of the above addresses, indicate whether the cache access will result in a hit or a miss.

Ans.

128 (Hex: 80)	**144 (Hex: 90)**	**2176 (Hex: 880)**	**2180 (Hex: 884)**	**128 (Hex: 80)**	**2176 (Hex: 880)**
00000 00010 000000	00000 00010 010000	00001 00010 000000	00001 00010 000100	00000 00010 000000	00001 00010 000000
MISS	**HIT**	**MISS**	**HIT**	**MISS**	**MISS**



Index	Valid	Tag	Data (Hex)	Dirty Bit
0	0	-	0	0
1	0	-	0	0
2	1	00001	BLOCK 22 WORD 0 - 63	0
3	0	-	0	0
4	0	-	0	0
5	0	-	0	0
6	0	-	0	0
7	0	-	0	0
8	0	-	0	0
9	0	-	0	0
10	0	-	0	0
11	0	-	0	0
12	0	-	0	0
13	0	-	0	0
14	0	-	0	0
15	0	-	0	0
16	0	-	0	0
17	0	-	0	0
18	0	-	0	0
19	0	-	0	0
20	0	-	0	0
21	0	-	0	0
22	0	-	0	0
23	0	-	0	0
24	0	-	0	0
25	0	-	0	0
26	0	-	0	0
27	0	-	0	0

2. For the above-mentioned problem, calculate and execute for 4way set associativity and fully associative mapping technique. For each technique randomly generate ten addresses and indicate whether the cache access will result in a hit or a miss. Assume block replacement policy as random.

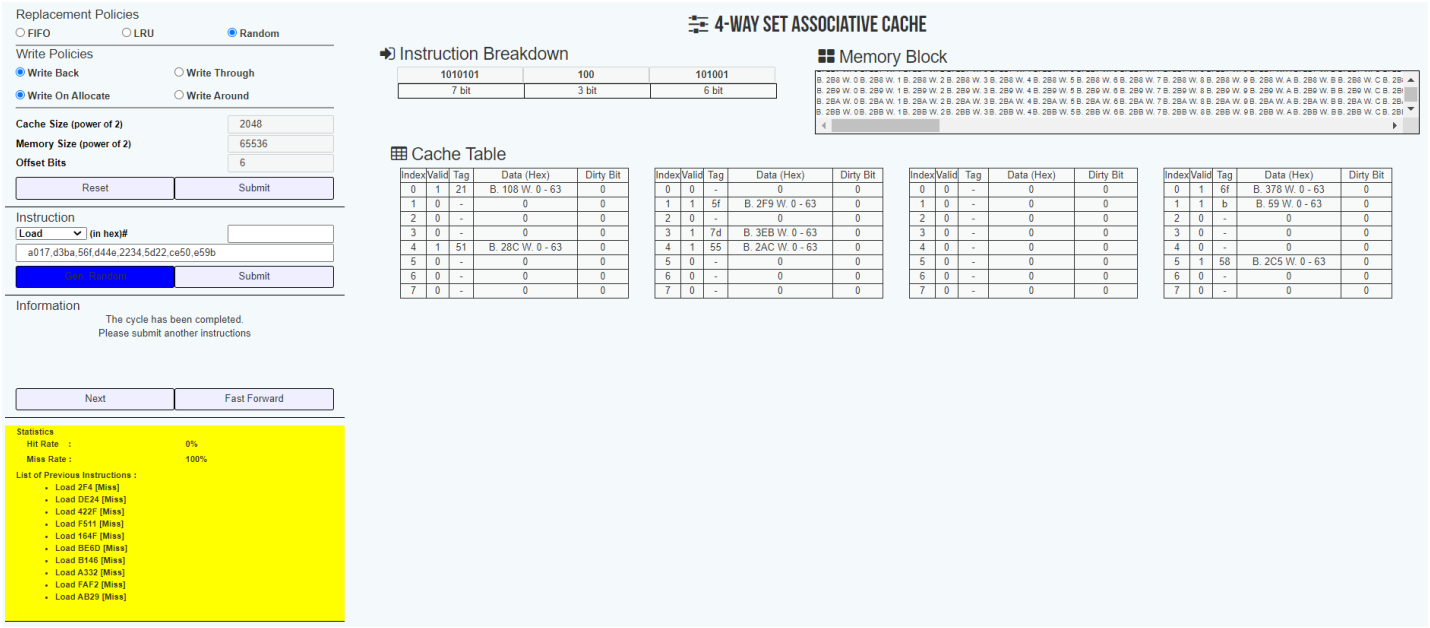
Ans.

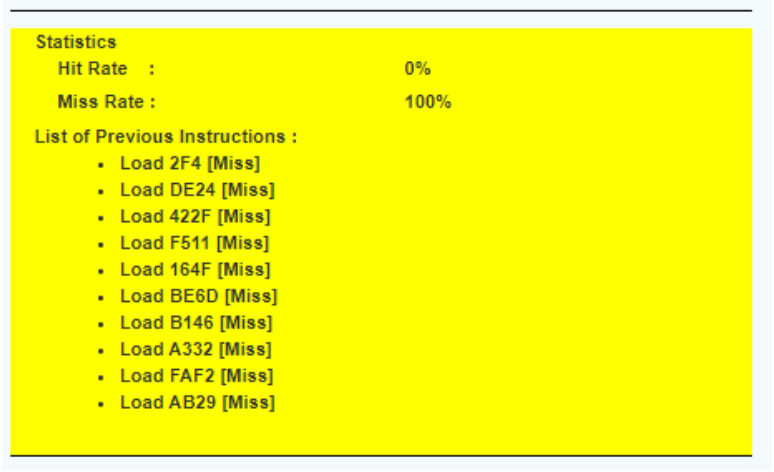
4-way set associative:

Index bits = Cache Size/ Set Size = 2^11 / 2^8 == 3 bits

Offset =6 bits

Tag = 16-6-3 = 7 bits



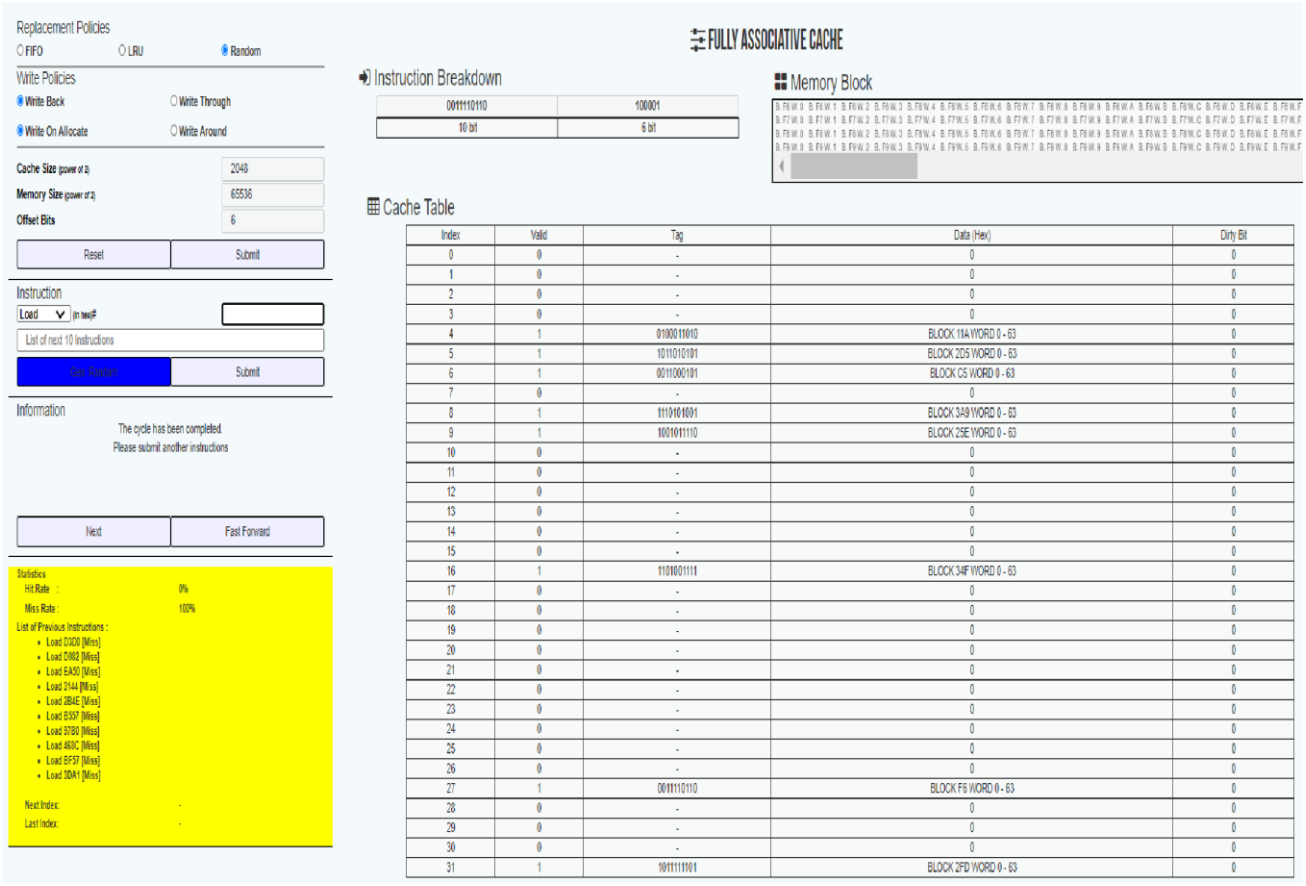


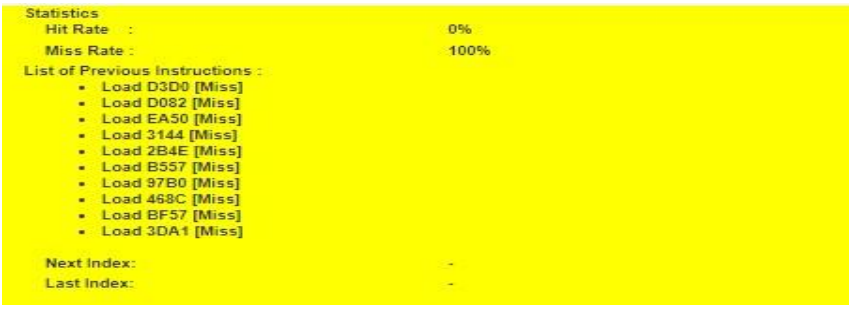
Fully associative:

Offset bits=6

Tag bits = 16-6 = 10

No index bits





Task3: Use Arm Simulator

Given a 'c' code convert it in its equivalent Arm Code and execute in ARM simulator.

1. x = (a + b) – c

.text

LDR R1, =a

LDR R2, =b

LDR R3, =c

LDR R1, [R1]

LDR R2, [R2]

LDR R3, [R3]

ADD R0, R1, R2

SUB R0, R0, R3

SWI 0X11

.data

a: .word 10

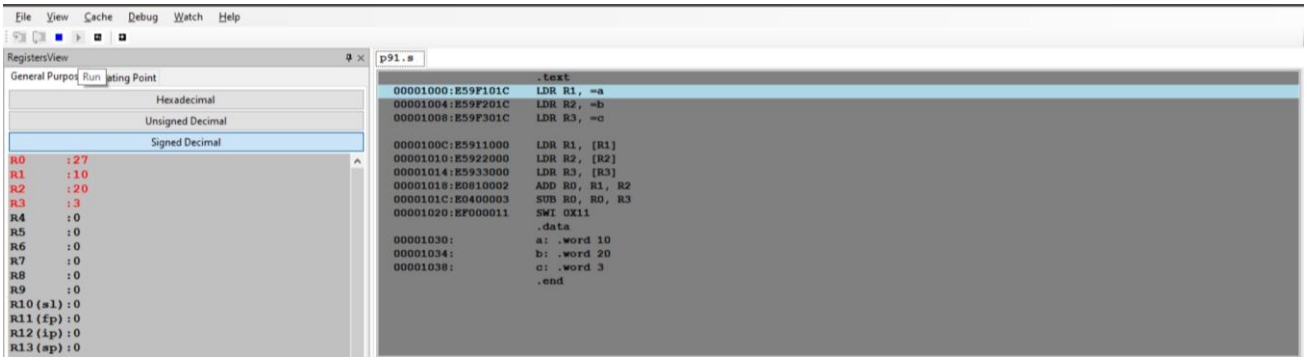
b: .word 20

c: .word 3

.end

Output:

R0: (10+20)-3=27



2) z = (a << 2) | (b & 15)

.text

MOV R0, #4

MOV R1, #7

MOV R3, R0, LSL #2

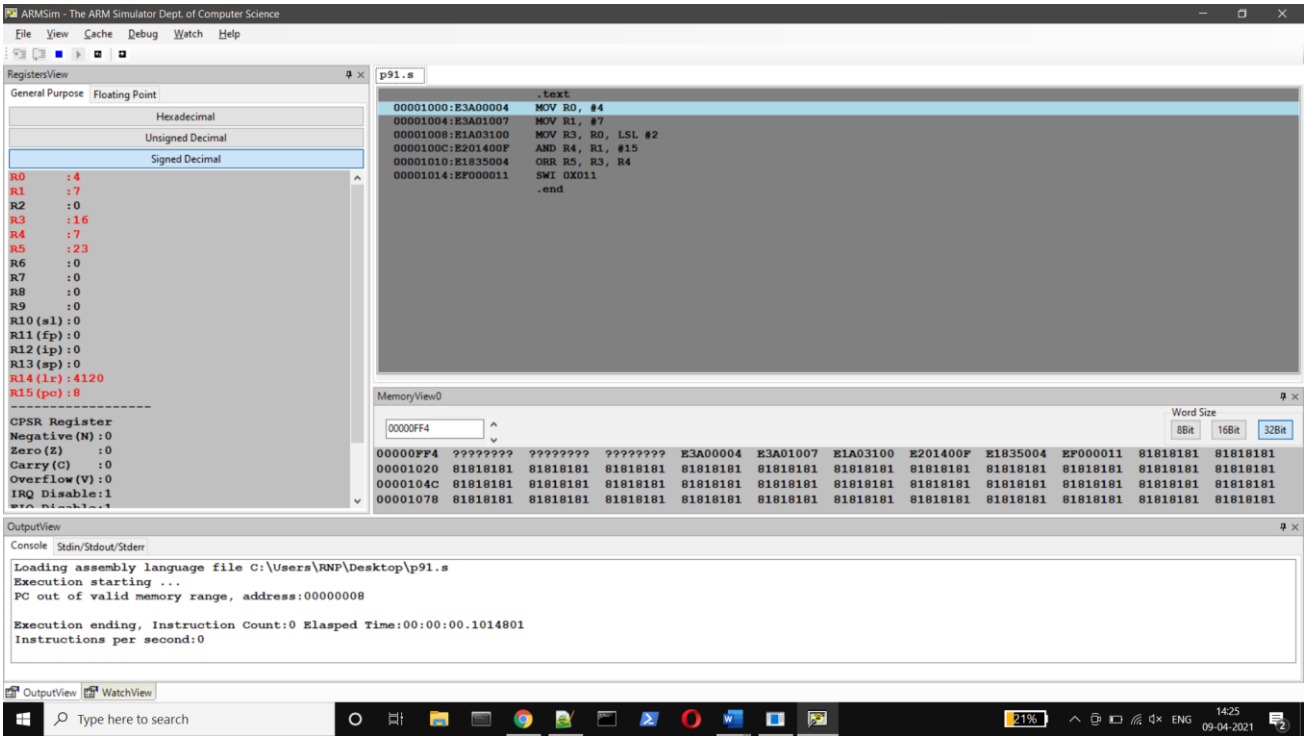
AND R4, R1, #15

ORR R5, R3, R4

SWI 0X011

.end

***Output**: (4<<2) | (7 & 15) = (0100 << 2) | (0111 & 1111) = 10000 | 0111 = 10111 = 23*


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 :4
R1 :7
R2 :0
R3 :16
R4 :7
R5 :23
R6 :0
R7 :0
R8 :0
R9 :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):0
R14(lr):4120
R15(pc):8
CPSR Register
Negative(N):0
Zero(Z) :0
Carry(C) :0
Overflow(V):0
IRQ Disable:1
p91.s
.text
00001000:E3A00004 MOV R0, #4
00001004:E3A01007 MOV R1, #7
00001008:E1A03100 MOV R3, R0, LSL #2
0000100C:E201400F AND R4, R1, #15
00001010:E1835004 ORR R5, R3, R4
00001014:EF000011 SWI 0X011
.end
MemoryView0
00000FF4
Word Size
8Bit 16Bit 32Bit
00000FF4 ???????? ???????? ???????? E3A00004 E3A01007 E1A03100 E201400F E1835004 EF000011 81818181 81818181
00001020 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
0000104C 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
00001078 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
OutputView
Console Stdin/Stdout/Stderr
Loading assembly language file C:\Users\RNP\Desktop\p91.s
Execution starting ...
PC out of valid memory range, address:00000008
Execution ending, Instruction Count:0 Elasped Time:00:00:00.1014801
Instructions per second:0
OutputView WatchView
Type here to search
21%
ENG
14:25
09-04-2021