Report of Lab 2.8

Lalit Meena,2019CS50439

For this Lab 2 we are building microarchitecture for subset of arm instructions.

This lab involves putting together some stage 1 modules, to form a simple processor which can execute the following subset of ARM instructions with limited variants/features. {add, sub, cmp, mov, ldr, str, beq, bne, b} .Then in stage 3 ,we extend single cycle datapath(stage 2) to flexible and efficient clock period multicycle path.

In stage4 ,we have tested our design exhaustively and reported waves/test-cases for most DP instructions. Then in stage 5 ,we are adding one more features to our implementation which supports shift operations on register and immediate operand variations.In stage 6 ,we are supporting more DT instructions features. It include byte and half word transfers (signed and unsigned), auto increment/decrement with option of pre/post indexing.Then in stage 7 we are including all variations of multiply instructions with mul_acc module

In stage 8 Included remaining instructions and features -bl ,swi instructions ,full predication

Zip folder name – L2.8_2019CS50439

Submission folder, **L2.8_2019CS50439** contains-

1. **alu.vhd –** alu module design file
2. **data_mem.vhd -** data memory 128x32 module design file
3. **rfile.vhd -** design file for module register file 16x32
4. **others.vhd-**contains additional modules require for stage-2**(Instruction Decoder, Condition Checker)-updated for full predication**
5. **flags.vhd-** design file for module flags **(one line change)**
6. **multicycle.vhd-** SIMPLE multi CYCLE PROCESSOR
7. **shifter.**vhd- Shifter which support 4 shift types-LSL,LSR,ASR,ROR
8. **PM.**vhd- combination circuit path between the processor and memory
9. **Mul_acc.vhd-** module to perform multiply -accumulator computation
10. **package.vhd -**package to declare some types.-etc **(states added)run.do**

testbench files-

testbench_multicycle.vhd - testbench file for module simple multicycle processor

final TESTCASES EPWAVE PICS-ftestcase-1,2,3,4,5,6….

final TESTCASES files- ftestcasen.s, n is 1,2,…..

Report of Lab 2.8-lab 2 stage 8 description and test cases(also screenshots)

More details are in next pages –

stage 8 changes in main multicycle processor

there require appropriate adjustment,signals to workable main programme signals.

Number of states used inside main clocked process is states-(fetch,read_AB,arithm,MSTR,MLDR,w2RF,READC,RSHIFT,WB2RF,MULAC,MW2RF,SWIST,RETRN, ELAST).Also used state signal in testbench to show at which state running currently.

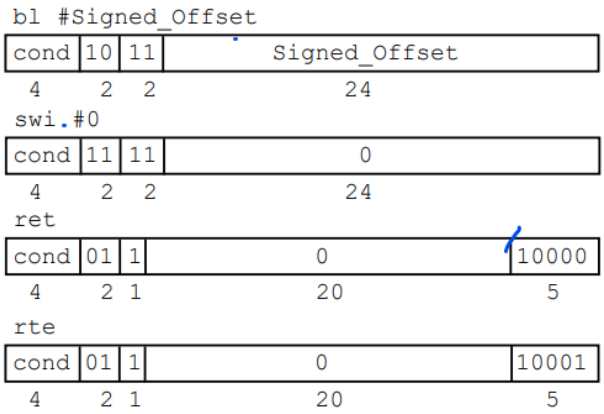
At this stage our design have only 14 states ,here we HAVE added tHREE new states –

swist-to branch swi related ISRs

retrn-to handle return type special cases -rte,ret

elast-to reach halted state when error produced ,program exit or access error related to priveledge mode

Format of new instructions.



Memory organization-DATA MEMORY (128x32)

byte adresses going from 0 to 511, that is from 0x00000000 to 0x000001FF

SUPERVISOR -system AREA-0x00000000 to 0x000000FF (0 to 255)

INPUT PORT- 0x00000070

ISR of reset can be placed at 0x00000020 –8

E6000011-RTE

ISR for SWI can be placed at 0x00000040-10

```
mov r1,#0X00000070 @port address
loop : ldr r0,[r1]
tst r0,#0X00000100 @checking status bit
beq loop
and r0,r0,#0X000000FF @ascii code
@rte
```

E3A01070

E5910000

E3100C01

0AFFFFFC

E2400030

E6000011

Instructions to branch to these addresses need to be placed at –

RESET-0x00000000 and SWI-0x00000008

For this branch case the byte offsets are (0x20- 0x08) and (0x40 - 0x08 - 0x08), i.e., 0x18 and 0x30

word offset – 0x06 , 0xc

USER AREA-0x00000100 to 0x000001FF (256 to 511)

User program starts from 0x00000100

Statck for user memory usage

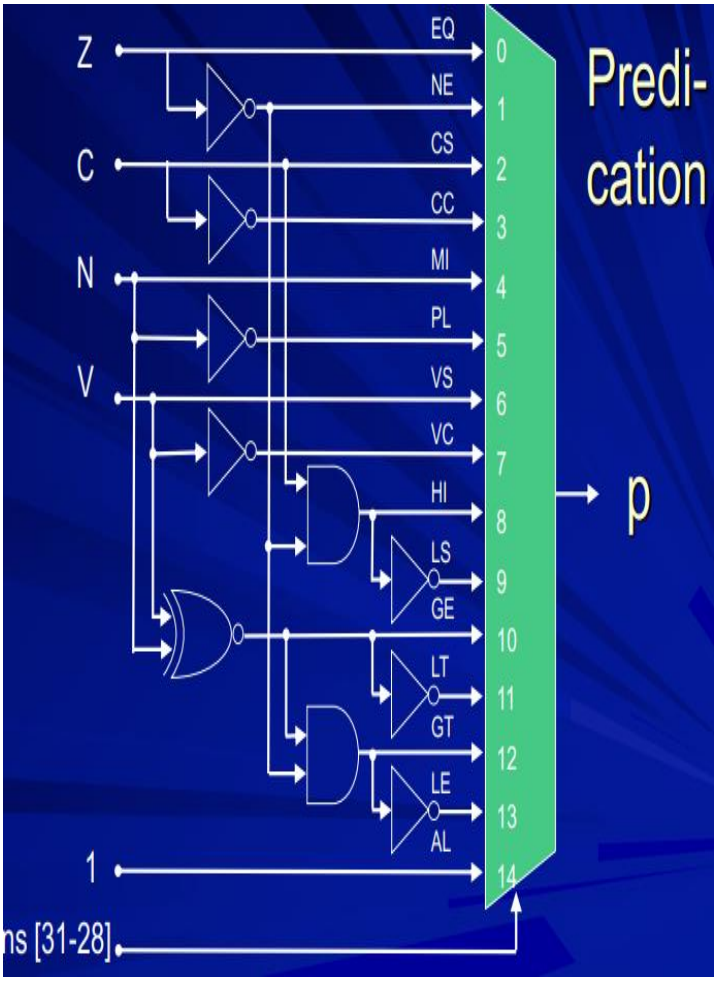
R13-sp - 0x000001FF

Full predication logic in checker module-

Checker module was previously tested in others.vhd.

Predication

cond	ins [31-28]	Flags
EQ	0 0 0 0	Z set
NE	0 0 0 1	Z clear
CS | HS	0 0 1 0	C set | higher or same
CC | LO	0 0 1 1	C clear | lower
MI	0 1 0 0	N set
PL	0 1 0 1	N clear
VS	0 1 1 0	V set
VC	0 1 1 1	V clear
HI	1 0 0 0	C set and Z clear
LS	1 0 0 1	C clear or Z set
GE	1 0 1 0	N = V
LT	1 0 1 1	N ≠ V
GT	1 1 0 0	Z clear and (N = V)
LE	1 1 0 1	Z set or (N ≠ V)
AL	1 1 1 0	ignored



Flags module is more tested here as also previously tested at each stage.

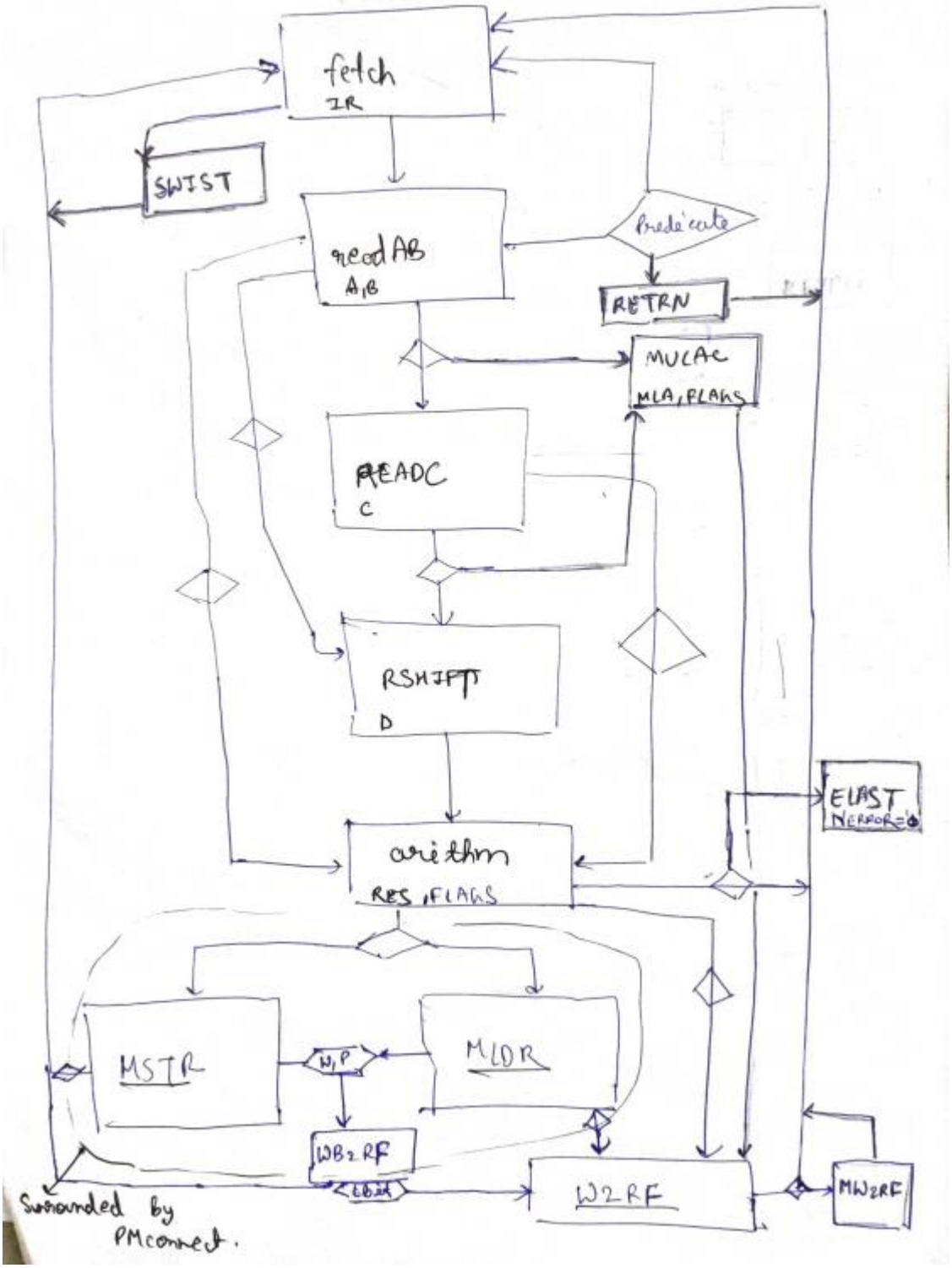
How DP instructions affect Flags C, V, Z, N

<table>
<tr><th rowspan="2">Instructions</th><th colspan="3">Effect on Flags</th></tr>
<tr><th>if S-bit = 0</th><th>if S-bit = 1 and
no shift/rotate</th><th>if S-bit = 1 and
shift/rotate is there</th></tr>
<tr><td>add, sub, rsb,
adc, sbc, rsc</td><td>No flags are affected</td><td colspan="2">All 4 flags are affected, ALU carry is used</td></tr>
<tr><td>cmp, cmn</td><td colspan="3">All 4 flags are affected, ALU carry is used</td></tr>
<tr><td>and, orr, xor, bic,
mov, mvn</td><td>No flags are affected</td><td>Only Z and N are
affected</td><td rowspan="2">C, Z and N are affected
shift/rotate carry is used</td></tr>
<tr><td>tst, teq</td><td colspan="2">Only Z and N are affected</td></tr>
</table>

Flags are setup exhaustively to handle shifter,multiply,carry etc all type variations.

Ref - 8.2 (16 Marks)

ASM CHART.



Testcases

simulated with help of testbench_multicycle.vhd-

state change like flags and registers at end of program or step can be checked by signal like A,B,C,D,FLAGS, IR,PC,RES,state,minstype ,MWetc.

Input-port testing

Here main things to look – mode signal and input-testbench related parameters(inpt-testbench setting signal to set input potrt ,indata(status+byte data).As in testbench combination of two are setting input port location in data memory.

After changing the input data, the status signal (9th bit) should become 1 for several clock cycles and then become 0 till the next data change

To handle it we **multiplexed signals** from **PMConnect and these testbench** to go in memory.

```
entity processor is
 port (
 clock, reset : in std_logic;
 INPT : in std_logic;
 INDATA : IN WORD
 );
end processor;
```

Correctness can be justified by values of registers read after instructions,etcAlso added extra mov instructions to read and show value of B,RES OR 2nd operand.

For errors check d-state(elast) .

We can observe number of cycles etc,also used state signal to show which state currently

Below are related testcases-which covers cases told to implement-

Read from input-storing memory

From User mode try system area access

BL and ret type

Full predication capability

Flags -sbit

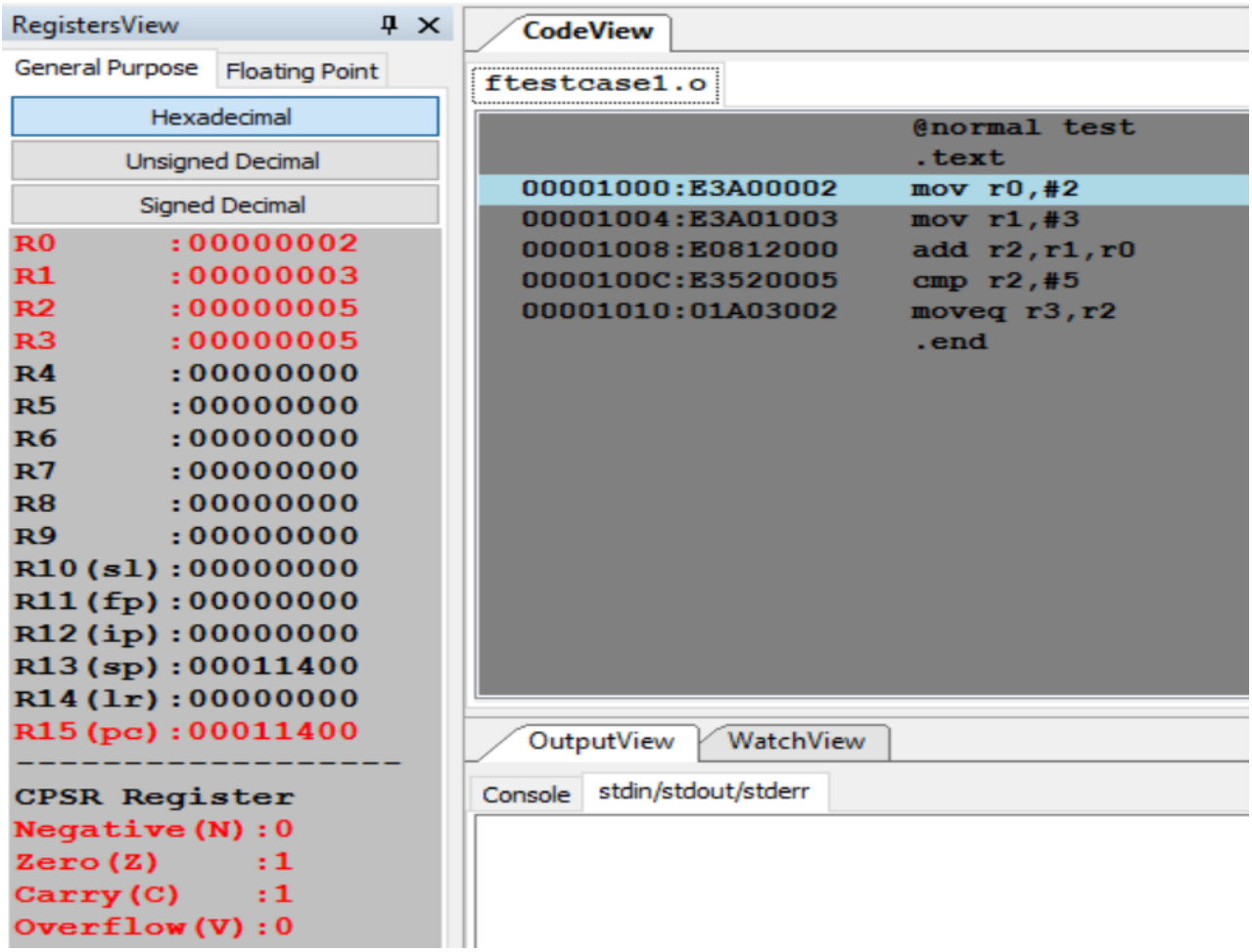
Each testcase was little explained and also ownself explanatory.

Take help of **run time limit** on simulator for taking required PC increments.For clock related testbench signals .

TESTCASES SEPARATE EPWAVE PICS and assembly files CAN also BE FOUND IN SUBMITTED FOLDER.

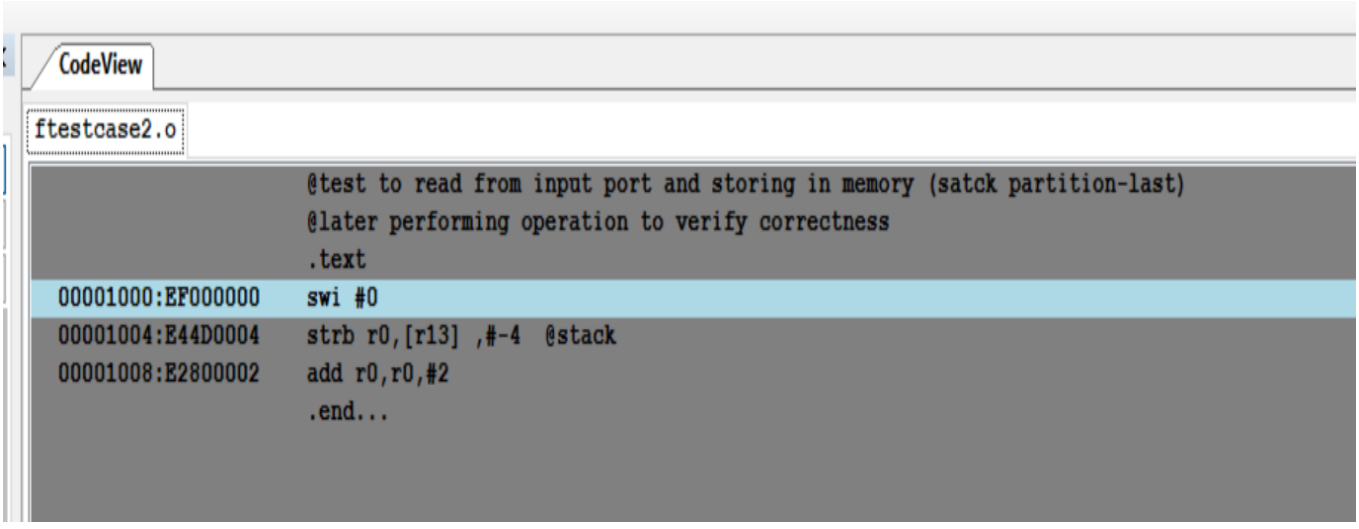
Ftestcase1

Compare ftestcase1.jpg with testcase3.jpg,only initial 2 reset related instructions are extra.



Ftestcase2

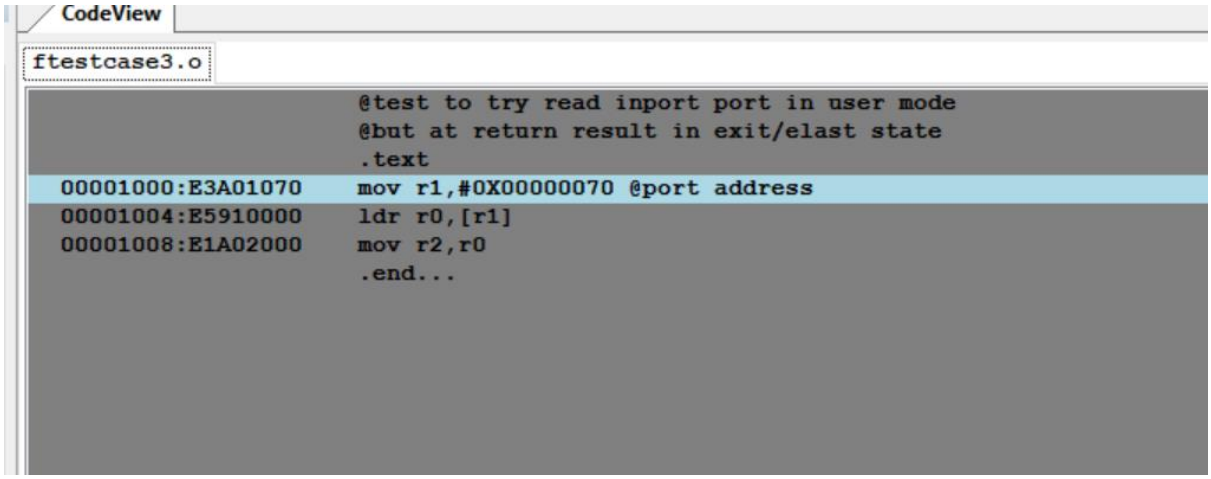
VERIFY 0x32 -2 ascii code input byte by seeing last instruction A register

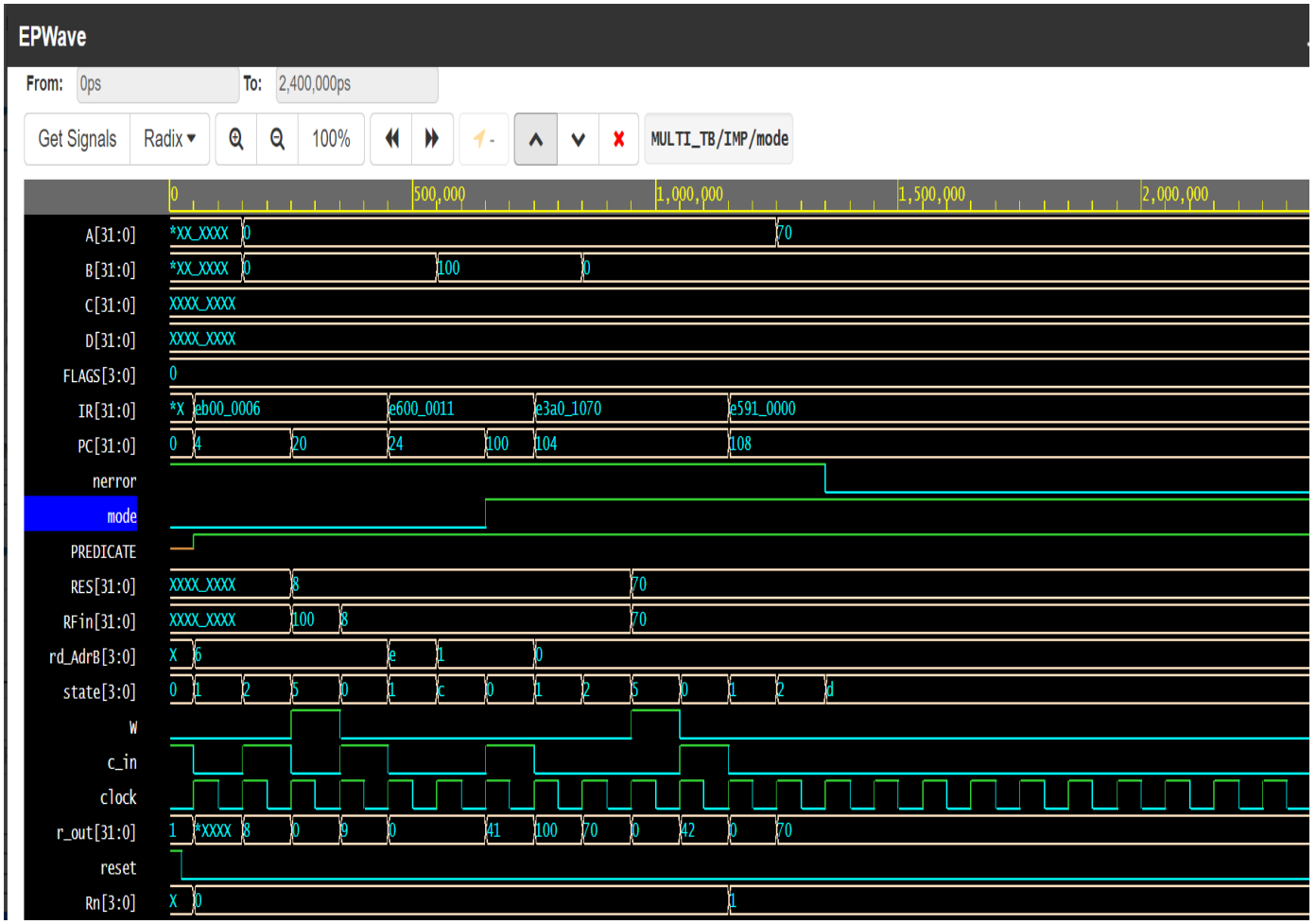


Ftestcase3

When trying to access supervisor partitioned while in user mode-

Program halt by going to error state(d-elast ,nerror = ‘1’)

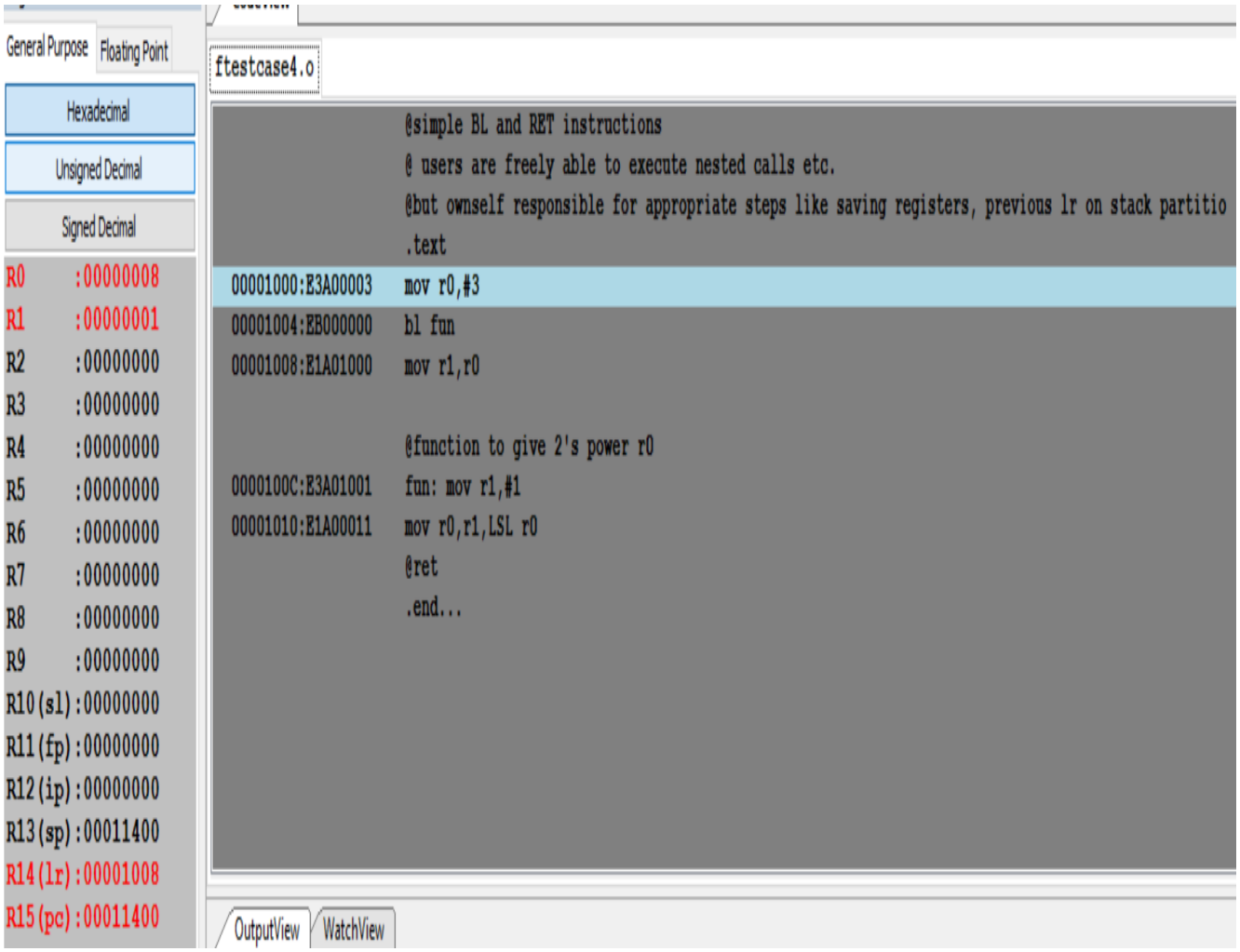




Ftestcase4-

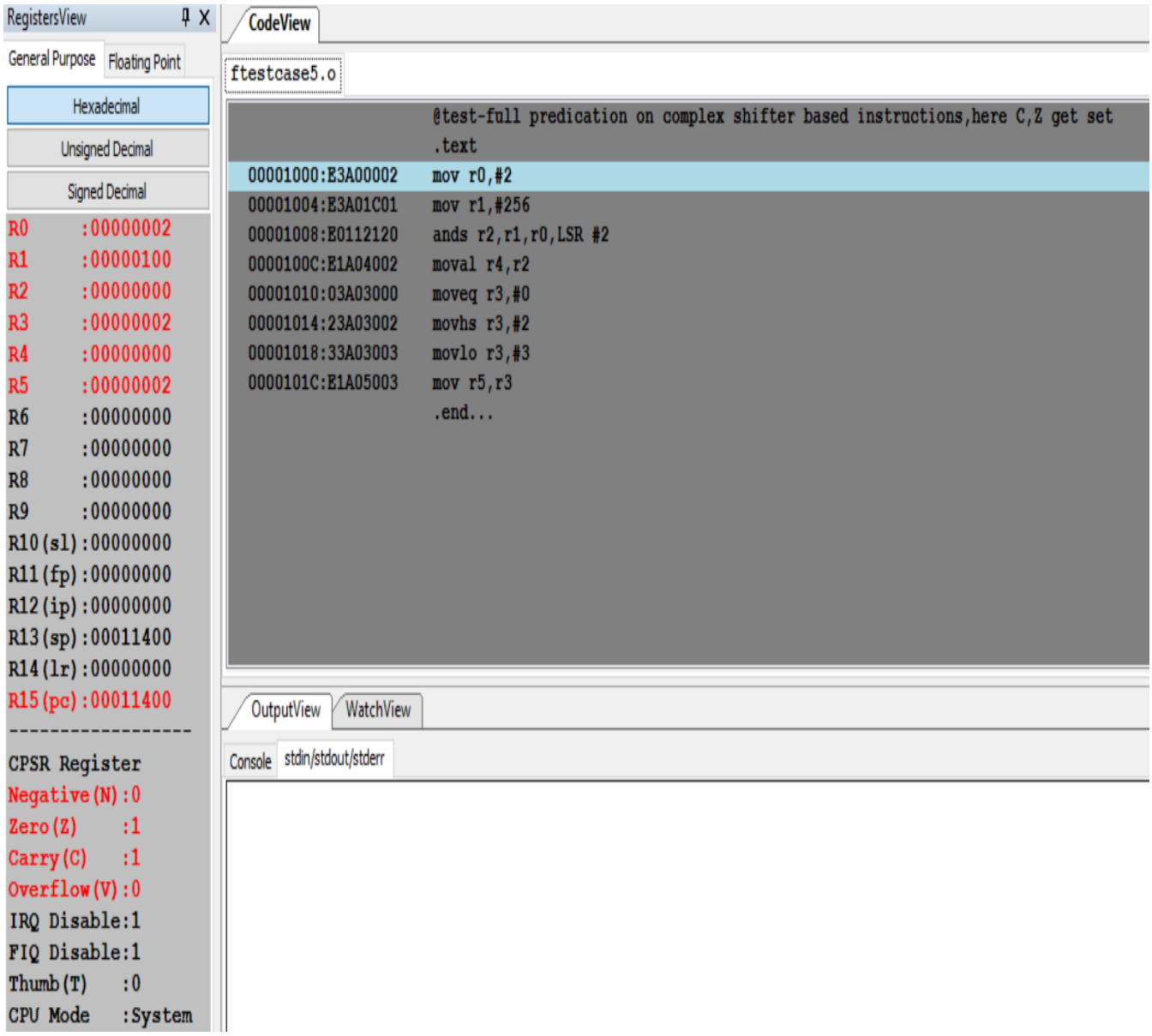
BL and ret type implemenatation with simple 2’s power function-8 result

users are freely able to execute nested calls etc.but ownself responsible for appropriate steps like saving registers, previous lr on stack partition-r13



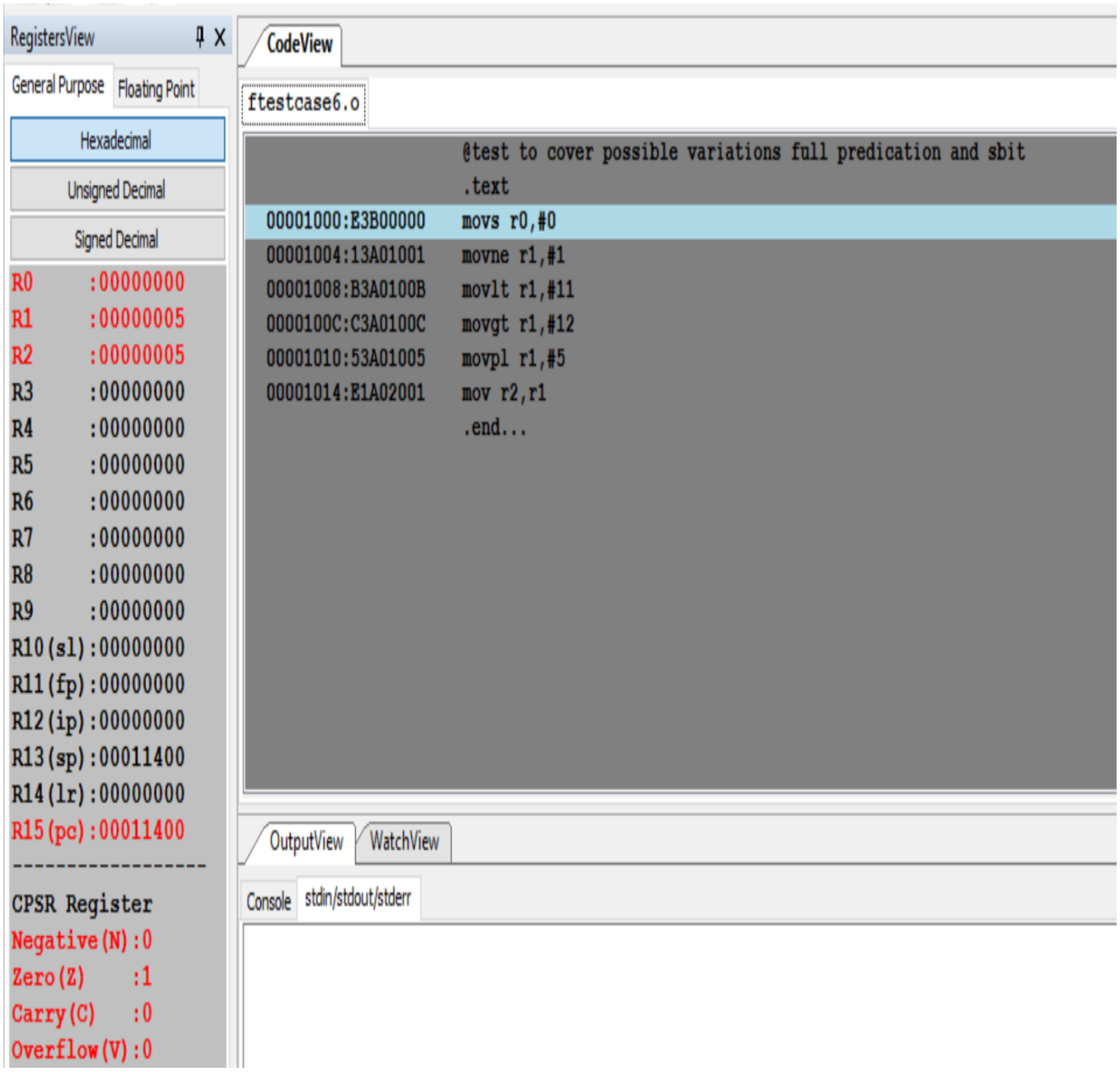
Ftestcase5

test-full predication on complex shifter based instructions,here C,Z get set



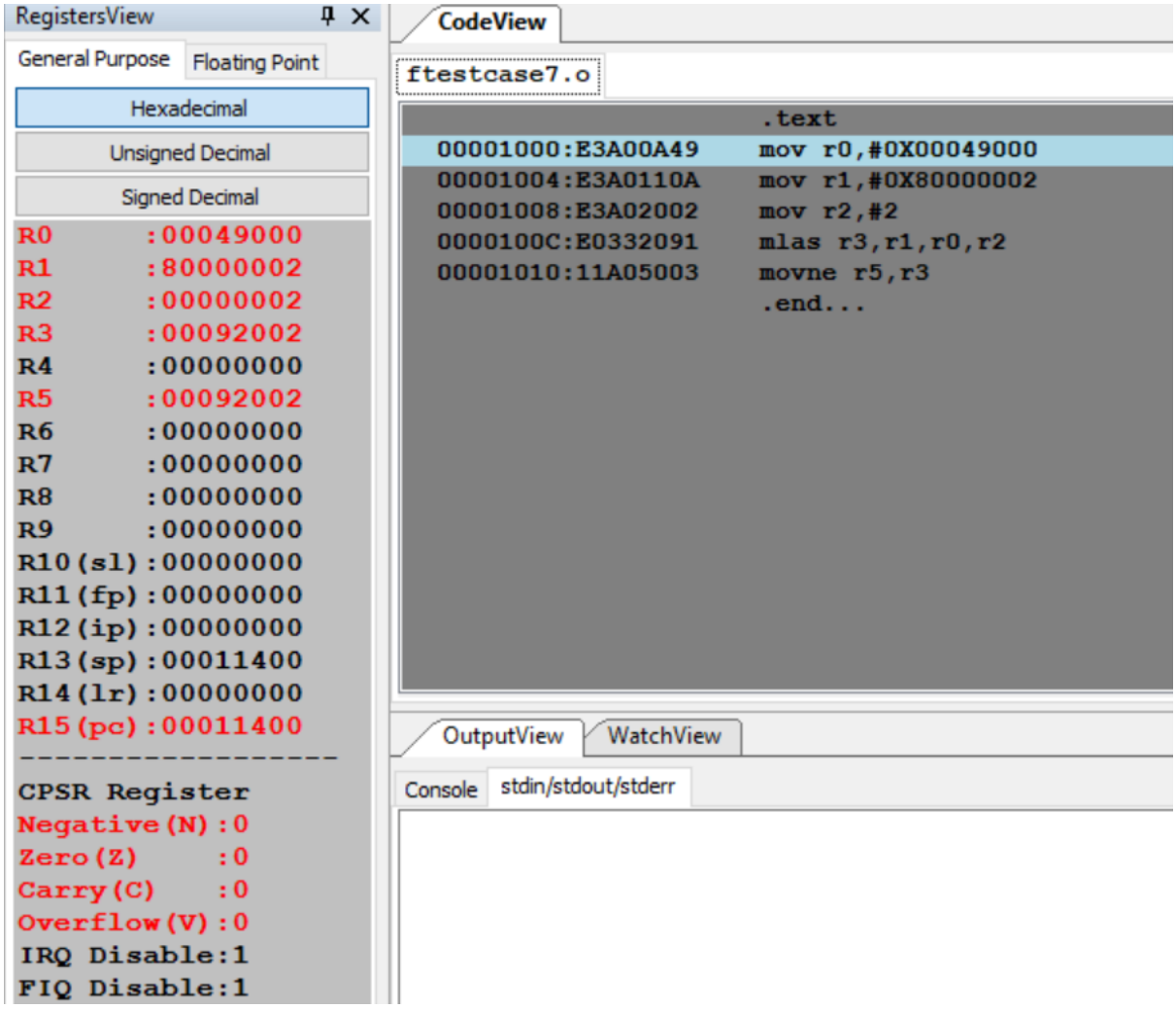
Ftestcase6

test to cover possible variations full predication and sbit



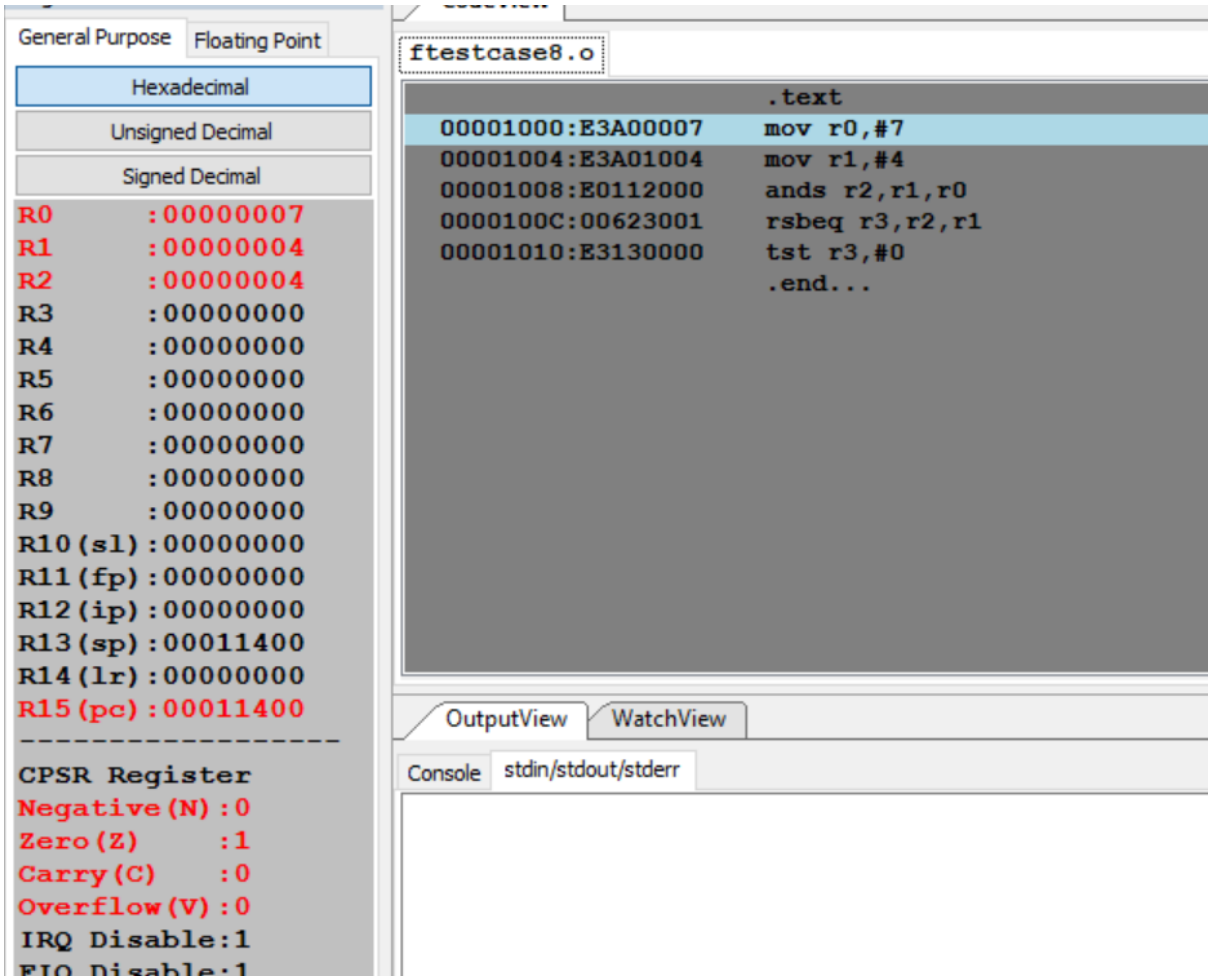
Ftestcase7

Flags and predication related variations



Ftestcase8

Flags and predication related variations



Thank you!