Team #8

Names	Sec	B.N.
Hossam Ahmed	**1**	**15**
Sarah Mohamed Ahmed Lotfy	**1**	**24**
Nada Adel	**2**	**29**
Nourhan Gamal	**2**	**33**

What is not working in our processor?

Memory cash is implemented but in individual module not merged to the system.

It is found in the Memory system folder with its .do file and it worked correctly.

One operand

Without forwarding & hazard & flushing units

Instruction	**Result from simulation**	**Correct results**
NOT R1	R1 = **FFFFFFFF**	R1 = **FFFFFFFF**
NOP	No change	No change
inc R1	R1 = **1**	R1 = **00000000**
IN R1	R1= **5**	R1 = **5**
IN R2	R2= **10**	R2= **10**
NOT R2	R2 = **FFFFFFFF**	R2 = **FFFFFFEF**
inc R1	R1 = **6**	R1 = **6**
Dec R2	R2 = **0000000F**	R2 = **FFFFFFEE**
Out R1	Port contains **5**	Port should contain **6**
OUT R2	Port contains **FFFFFFFF**	Port should contain **FFFFFFEE**

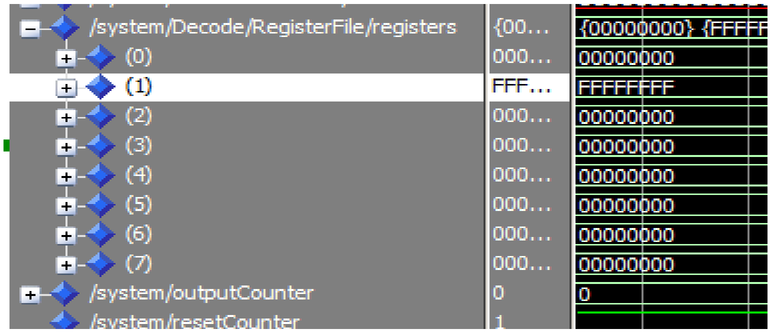
One operand : Solve by adding NOPs

Instruction	Hazard	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
NOT R1		F	D	E	M	W																	
NOP			F	D	E	M	W																
NOP				F	D	E	M	W															
INC R1					F	D	E	M	W														
IN R1	WAW R1 Data Hazard					F	D	E	M	W													
IN R2							F	D	E	M	W												
NOP								F	D	E	M	W											
NOP									F	D	E	M	W										
NOT R2	WAW R2 Data Haza									F	D	E	M	W									
inc R1											F	D	E	M	W								

NOP												F	D	E	M	W							
Dec R2	RAW R2 Data Haza												F	D	E	M	W						
Out R1	RAW R1 Data Haza													F	D	E	M	W					
NOP															F	D	E	M	W				
OUT R2	RAW R2 Data Haza															F	D	E	M	W			

The following waves shows the register file without forwarding unit & hazard detection & flush

[1] NOT R1

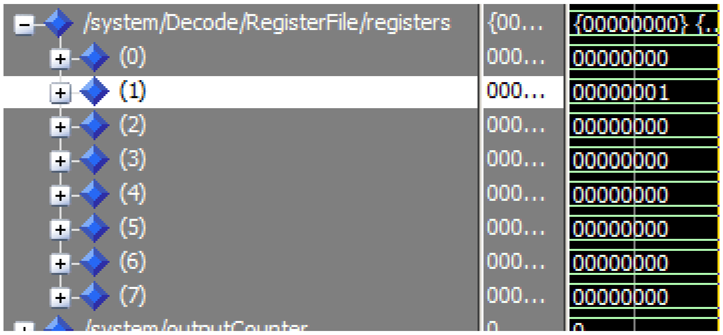


/system/Decode/RegisterFile/registers	{00…	{00000000} {FFFFF
(0)	000…	00000000
(1)	FFF…	FFFFFFFF
(2)	000…	00000000
(3)	000…	00000000
(4)	000…	00000000
(5)	000…	00000000
(6)	000…	00000000
(7)	000…	00000000
/system/outputCounter	0	0
/system/resetCounter	1	

[2]NOP

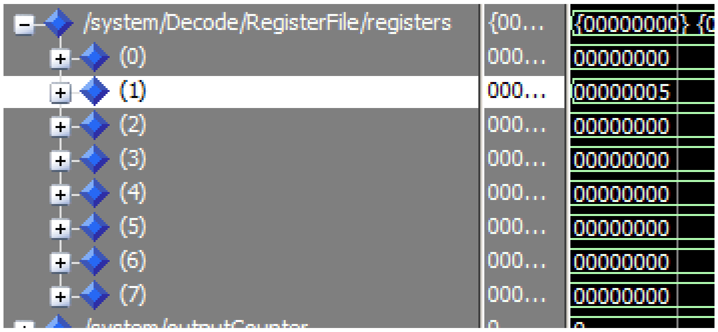
No change

[3] inc R1



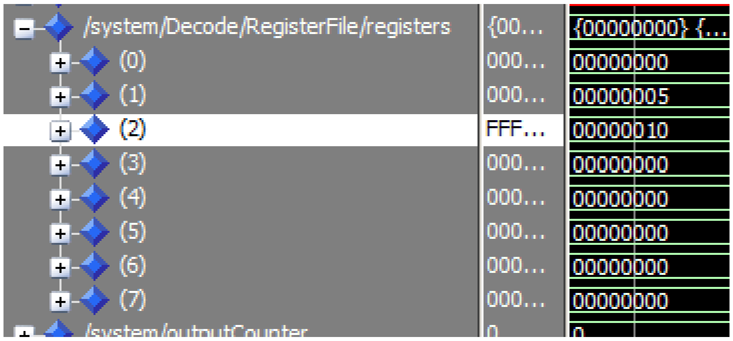
/system/Decode/RegisterFile/registers	{00…	{00000000} {…
(0)	000…	00000000
(1)	000…	00000001
(2)	000…	00000000
(3)	000…	00000000
(4)	000…	00000000
(5)	000…	00000000
(6)	000…	00000000
(7)	000…	00000000
/system/outputCounter	0	0

[4] in R1



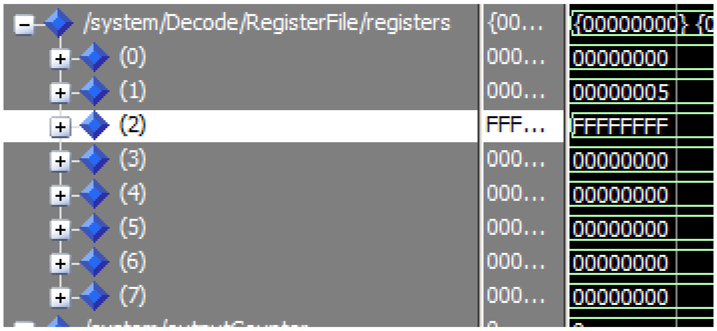
```
/system/Decode/RegisterFile/registers   {00...   {00000000} {0
  (0)                                   000...   00000000
  (1)                                   000...   00000005
  (2)                                   000...   00000000
  (3)                                   000...   00000000
  (4)                                   000...   00000000
  (5)                                   000...   00000000
  (6)                                   000...   00000000
  (7)                                   000...   00000000
/system/outputCounter                   0        0
```

[5] in R2



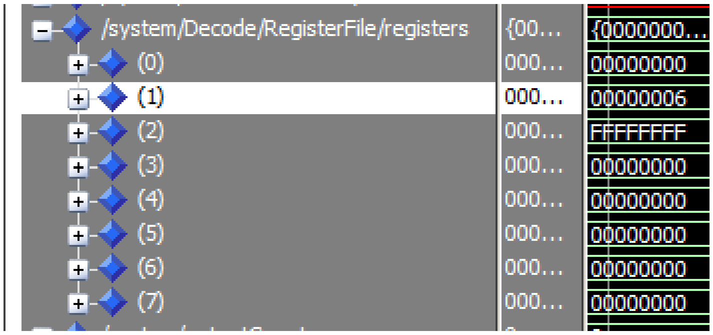
```
/system/Decode/RegisterFile/registers   {00...   {00000000} {...
  (0)                                   000...   00000000
  (1)                                   000...   00000005
  (2)                                   FFF...   00000010
  (3)                                   000...   00000000
  (4)                                   000...   00000000
  (5)                                   000...   00000000
  (6)                                   000...   00000000
  (7)                                   000...   00000000
/system/outputCounter                   0        0
```

[6] NOT R2



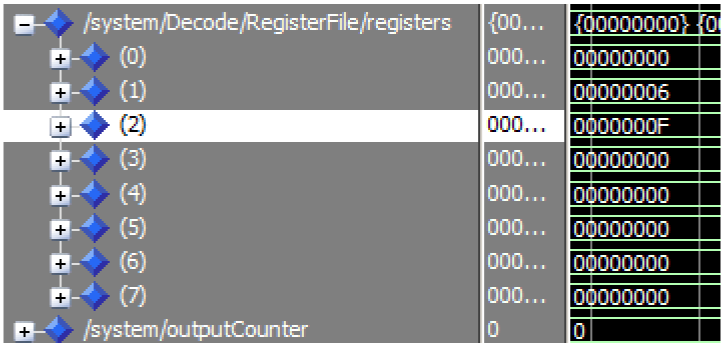
```
/system/Decode/RegisterFile/registers   {00...   {00000000} {0
  (0)                                   000...   00000000
  (1)                                   000...   00000005
  (2)                                   FFF...   FFFFFFFF
  (3)                                   000...   00000000
  (4)                                   000...   00000000
  (5)                                   000...   00000000
  (6)                                   000...   00000000
  (7)                                   000...   00000000
/system/outputCounter                   0        0
```

[7] inc R1



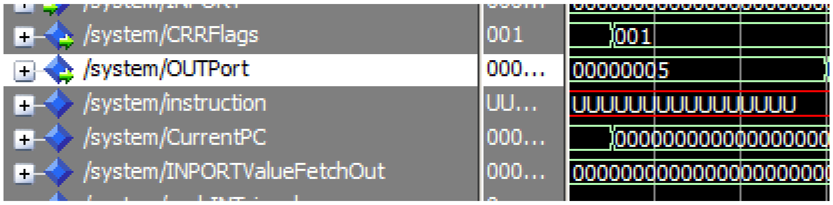
```
/system/Decode/RegisterFile/registers   {00...   {0000000...
  (0)                                   000...   00000000
  (1)                                   000...   00000006
  (2)                                   000...   FFFFFFFF
  (3)                                   000...   00000000
  (4)                                   000...   00000000
  (5)                                   000...   00000000
  (6)                                   000...   00000000
  (7)                                   000...   00000000
```

[8] Dec R2



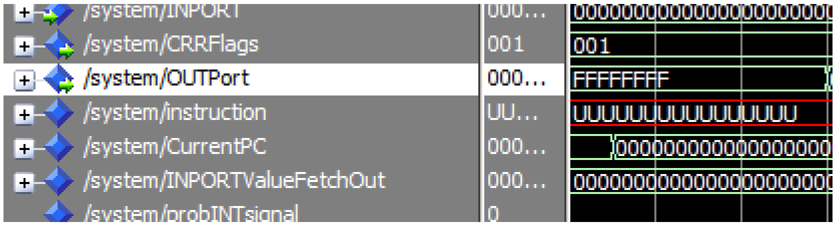
```
/system/Decode/RegisterFile/registers   {00...   {00000000} {0(
  (0)                                   000...   00000000
  (1)                                   000...   00000006
  (2)                                   000...   0000000F
  (3)                                   000...   00000000
  (4)                                   000...   00000000
  (5)                                   000...   00000000
  (6)                                   000...   00000000
  (7)                                   000...   00000000
/system/outputCounter                   0        0
```

[9] out R1 (Here is the value of OUtPort)



```
/system/CRRFlags                 001     001
/system/OUTPort                  000...  00000005
/system/instruction              UU...   UUUUUUUUUUUUUUUU
/system/CurrentPC                000...  000000000000000000
/system/INPORTValueFetchOut      000...  0000000000000000000000
```

[10] out R2



```
/system/INPORT                   000...  0000000000000000000000
/system/CRRFlags                 001     001
/system/OUTPort                  000...  FFFFFFFF
/system/instruction              UU...   UUUUUUUUUUUUUUUU
/system/CurrentPC                000...  000000000000000000
/system/INPORTValueFetchOut      000...  0000000000000000000000
/system/probINTsignal            0
```

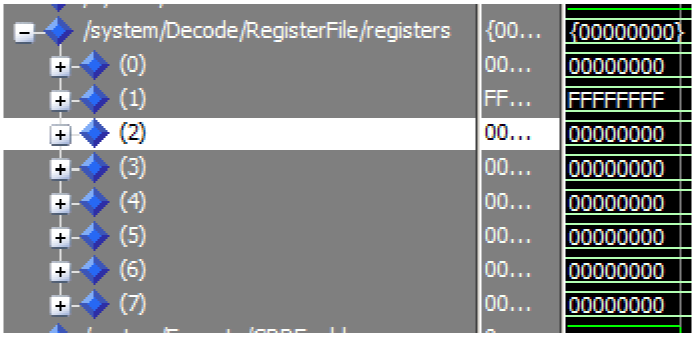
Here is the correct result with forwarding unit only

All instructions works CORRECTLY and give the right results with no hazards because all hazards are data hazards that are handled by forwarding unit.

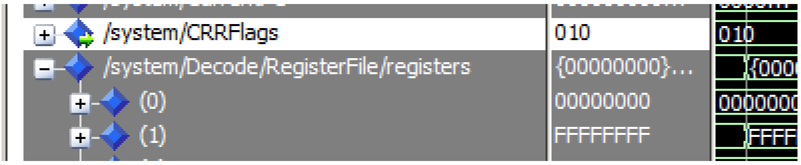
The following wave shows the register file

Z flag is CRRFlag(0)
N flag is CRRFlag(1)
C flag is CRRFlag(2))

[1] NOT R1



```
/system/Decode/RegisterFile/registers   {00...   {00000000}
    (0)                                 00...    00000000
    (1)                                 FF...    FFFFFFFF
    (2)                                 00...    00000000
    (3)                                 00...    00000000
    (4)                                 00...    00000000
    (5)                                 00...    00000000
    (6)                                 00...    00000000
    (7)                                 00...    00000000
```

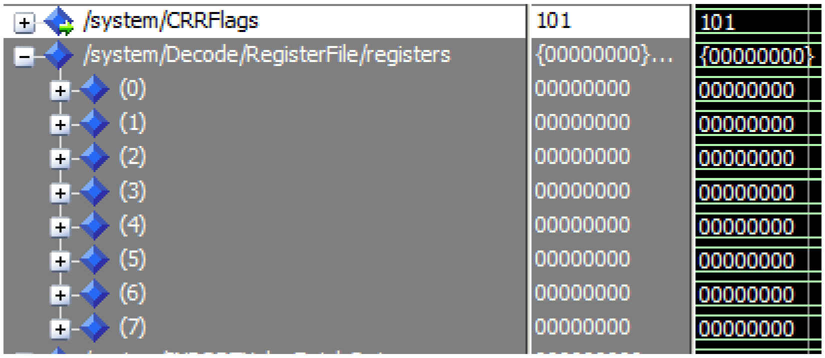


```
/system/CRRFlags                        010             010
/system/Decode/RegisterFile/registers   {00000000}...   {000
    (0)                                 00000000        0000000
    (1)                                 FFFFFFFF        FFFF
```

[2]NOP

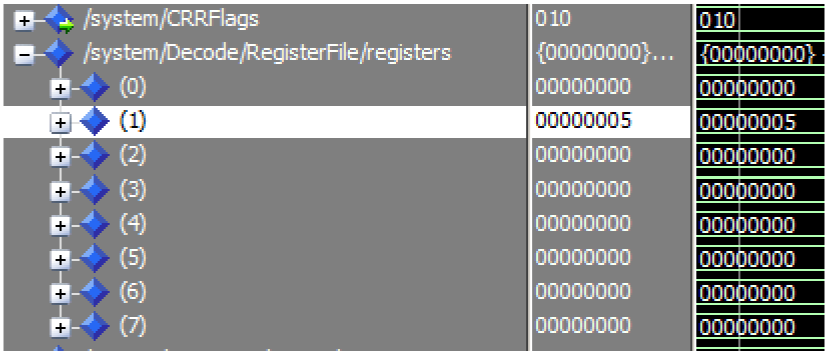
No change

[3] inc R1



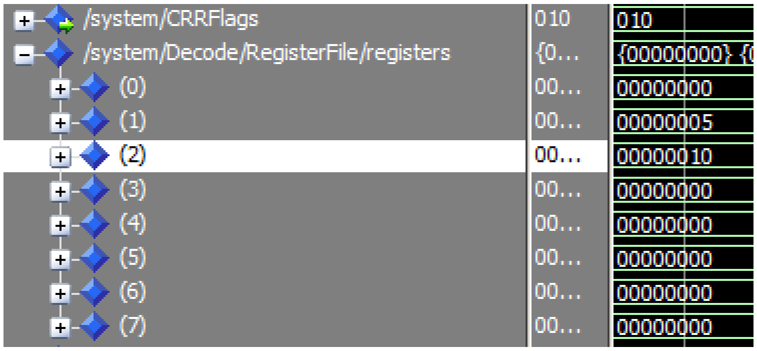
```
+ /system/CRRFlags                          101               101
- /system/Decode/RegisterFile/registers     {00000000}...     {00000000}
   + (0)                                    00000000          00000000
   + (1)                                    00000000          00000000
   + (2)                                    00000000          00000000
   + (3)                                    00000000          00000000
   + (4)                                    00000000          00000000
   + (5)                                    00000000          00000000
   + (6)                                    00000000          00000000
   + (7)                                    00000000          00000000
```

[4] in R1



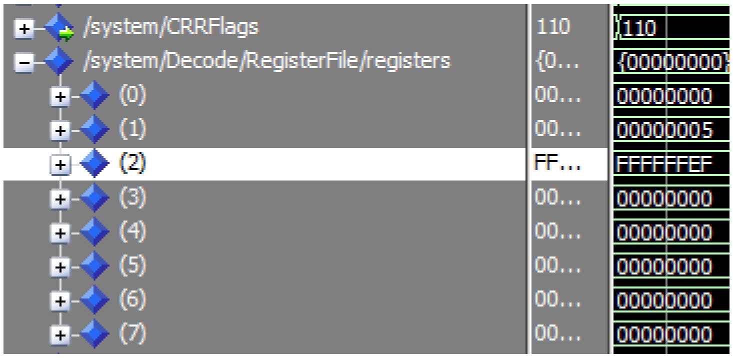
```
+ /system/CRRFlags                          010               010
- /system/Decode/RegisterFile/registers     {00000000}...     {00000000}
   + (0)                                    00000000          00000000
   + (1)                                    00000005          00000005
   + (2)                                    00000000          00000000
   + (3)                                    00000000          00000000
   + (4)                                    00000000          00000000
   + (5)                                    00000000          00000000
   + (6)                                    00000000          00000000
   + (7)                                    00000000          00000000
```

[5] in R2



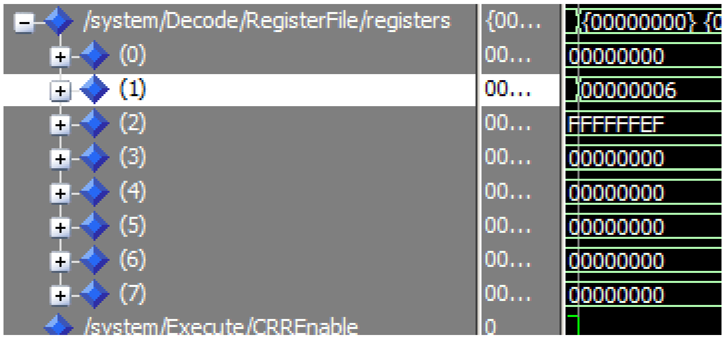
```
+ /system/CRRFlags                          010      010
- /system/Decode/RegisterFile/registers     {0...    {00000000} {(
   + (0)                                    00...    00000000
   + (1)                                    00...    00000005
   + (2)                                    00...    00000010
   + (3)                                    00...    00000000
   + (4)                                    00...    00000000
   + (5)                                    00...    00000000
   + (6)                                    00...    00000000
   + (7)                                    00...    00000000
```

[6] NOT R2



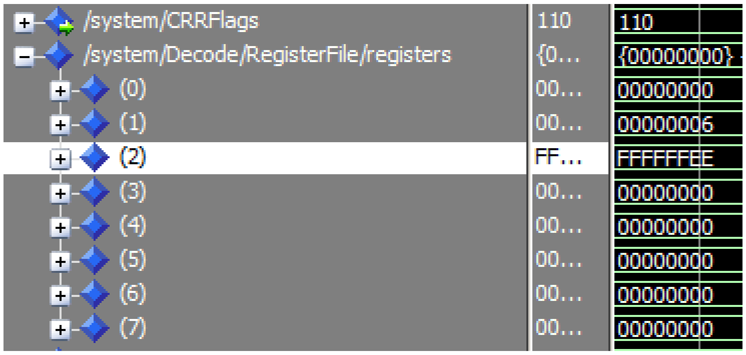
```
/system/CRRFlags                        110    110
/system/Decode/RegisterFile/registers   {0...  {00000000}
  (0)                                   00...  00000000
  (1)                                   00...  00000005
  (2)                                   FF...  FFFFFFEF
  (3)                                   00...  00000000
  (4)                                   00...  00000000
  (5)                                   00...  00000000
  (6)                                   00...  00000000
  (7)                                   00...  00000000
```

[7] inc R1



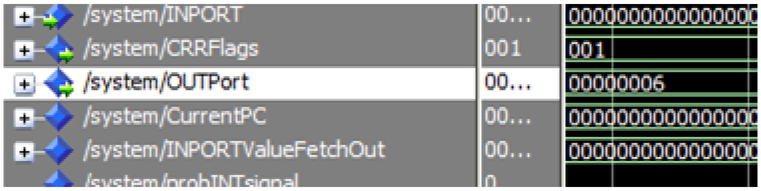
```
/system/Decode/RegisterFile/registers   {00...  {00000000} {0
  (0)                                   00...   00000000
  (1)                                   00...   00000006
  (2)                                   00...   FFFFFFEF
  (3)                                   00...   00000000
  (4)                                   00...   00000000
  (5)                                   00...   00000000
  (6)                                   00...   00000000
  (7)                                   00...   00000000
/system/Execute/CRREnable               0
```

[8] Dec R2



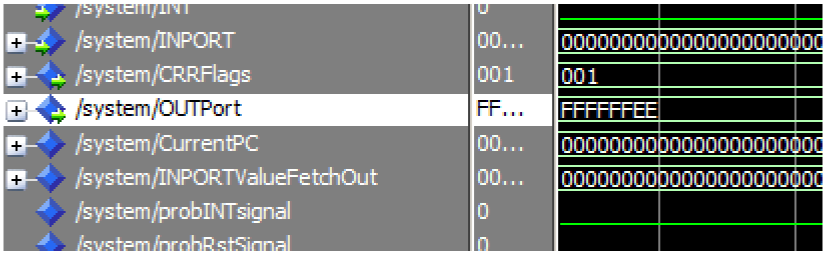
```
/system/CRRFlags                        110    110
/system/Decode/RegisterFile/registers   {0...  {00000000}
  (0)                                   00...  00000000
  (1)                                   00...  00000006
  (2)                                   FF...  FFFFFFEE
  (3)                                   00...  00000000
  (4)                                   00...  00000000
  (5)                                   00...  00000000
  (6)                                   00...  00000000
  (7)                                   00...  00000000
```

[9] out R1 (Here is the value of OUtPort)



```
/system/INPORT                  00...  0000000000000000
/system/CRRFlags                001    001
/system/OUTPort                 00...  00000006
/system/CurrentPC               00...  0000000000000000
/system/INPORTValueFetchOut     00...  0000000000000000
/system/probINTsignal           0
```

[10] out R2



/system/INT	0	
/system/INPORT	00...	0000000000000000000000
/system/CRRFlags	001	001
/system/OUTPort	FF...	FFFFFFEE
/system/CurrentPC	00...	0000000000000000000000
/system/INPORTValueFetchOut	00...	0000000000000000000000
/system/probINTsignal	0	
/system/probRstSignal	0	

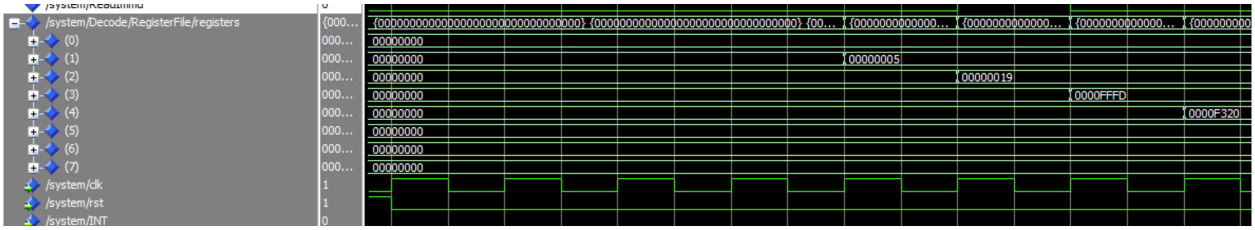
Two operand

- Without forward & hazard & flushing units

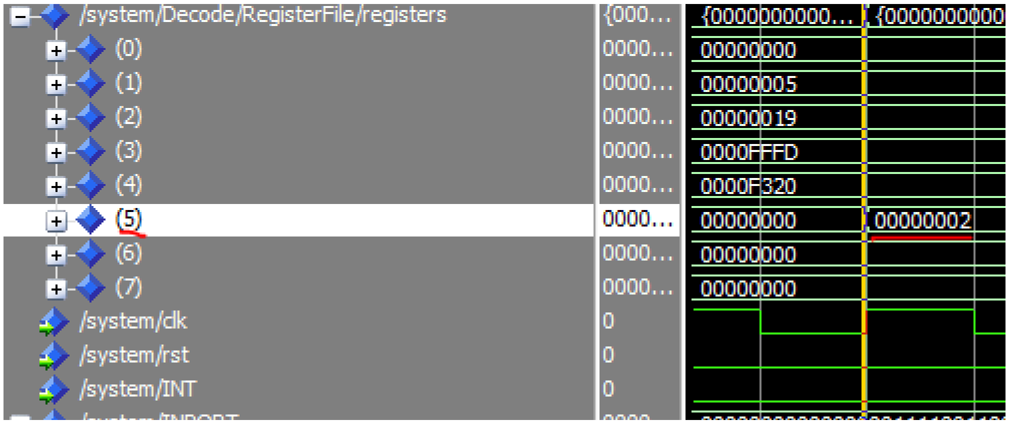
Instruction	Result from simulation	Correct results
IN R1	5	5
IN R2	19	19
IN R3	FFFD	FFFD
IN R4	F320	F320
IADD R3,R5,2	2	FFFF
ADD R1,R4,R4	F325	F325
SUB R5,R4,R6	FFFF0CE2	0CDA
AND R7,R6,R6	00000000	00000000
OR R2,R1,R1	1D	1D
SHL R2,2	64	64
SHR R2,3	3	0C
SWAP R2,R5	#R5=3 #R2=	#R5=0C #R2 =
ADD R5,R2,R2	5	1000B

The following screen shots from simulation shows the results without hazard detection unit ,forwad unit and flush

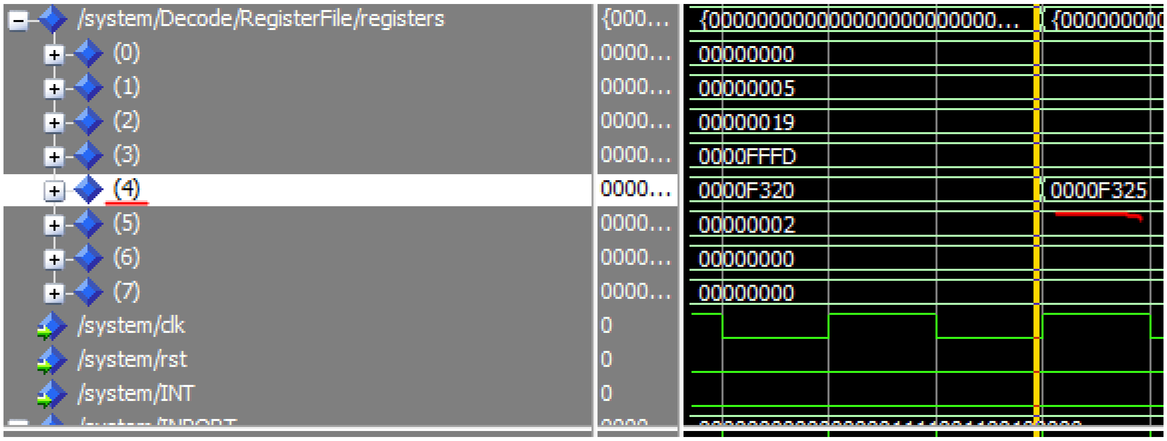
1. IN R1 , IN R2 , IN R3, INR4



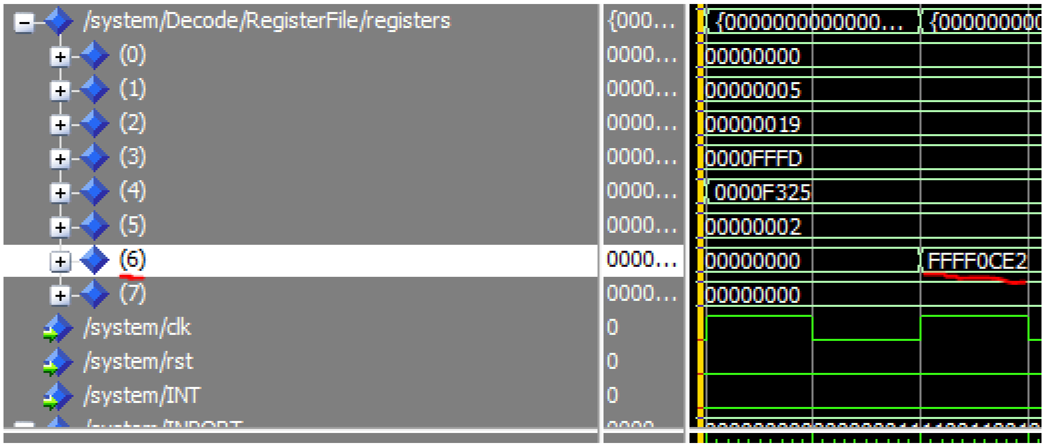
- IADD R3,R5,2



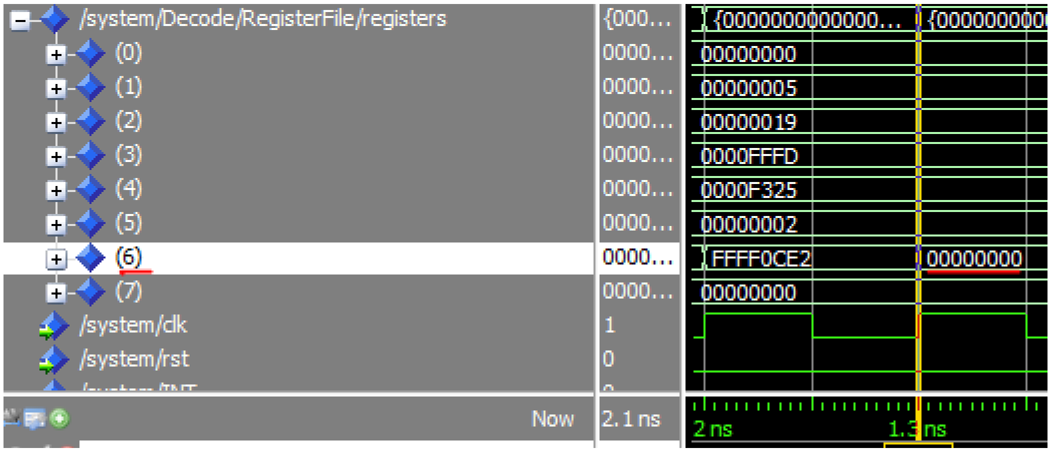
- ADD R1,R4,R4



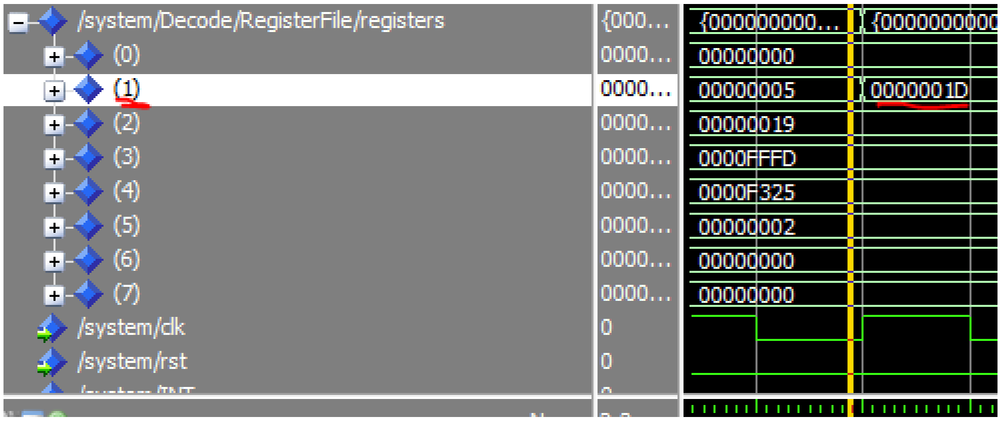
- SUB R5,R4,R6



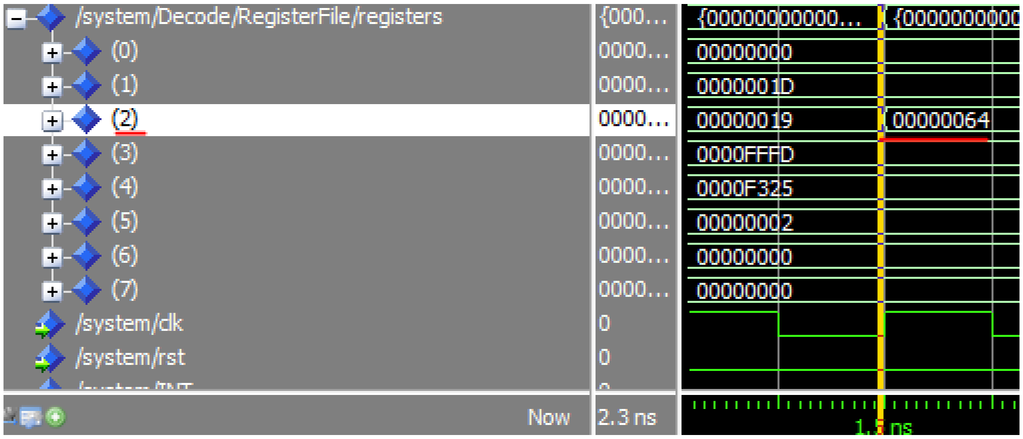
- AND R7,R6,R6



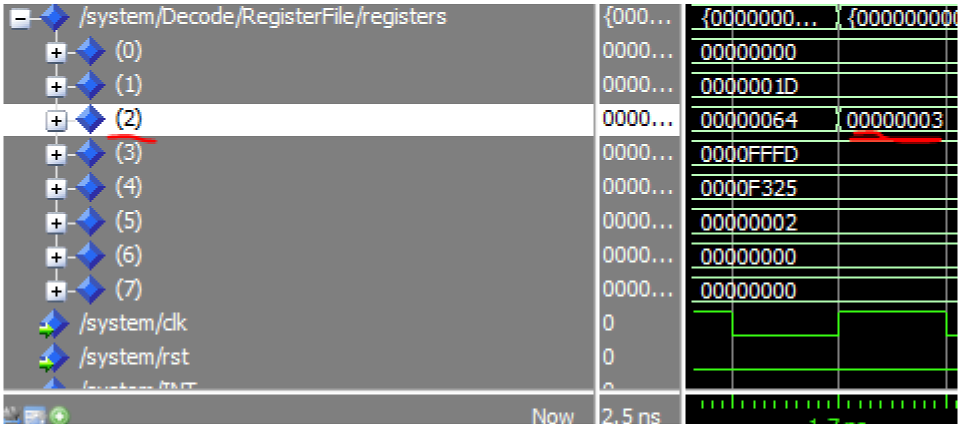
- OR   R2,R1,R1



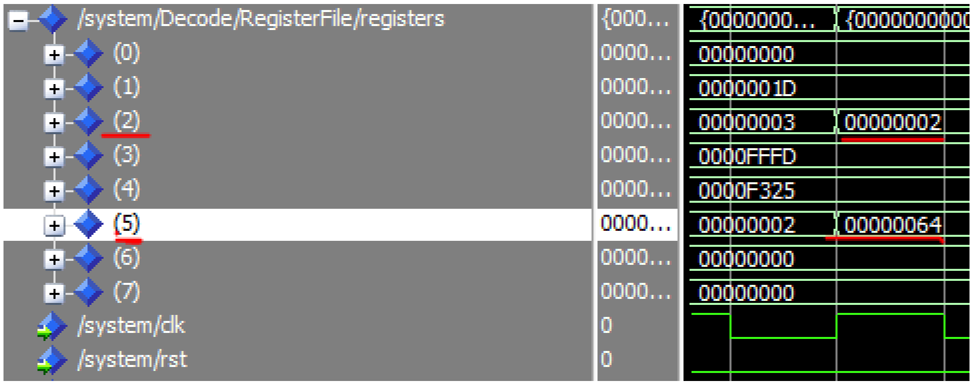
- SHL R2,2



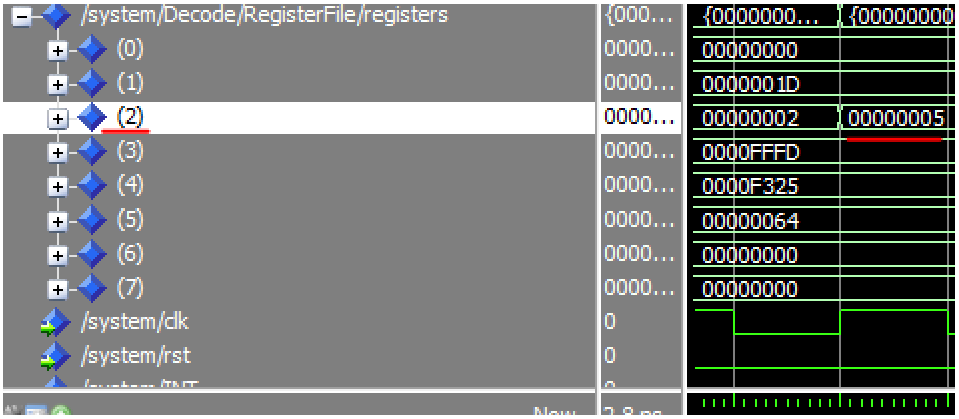
- SHR R2,3



- SWAP R2,R5



- ADD R5,R2,R2



- The following tabe shows the types of hazards happen (without forward unit, hazard detection and flush) and how can we solve it by NOP

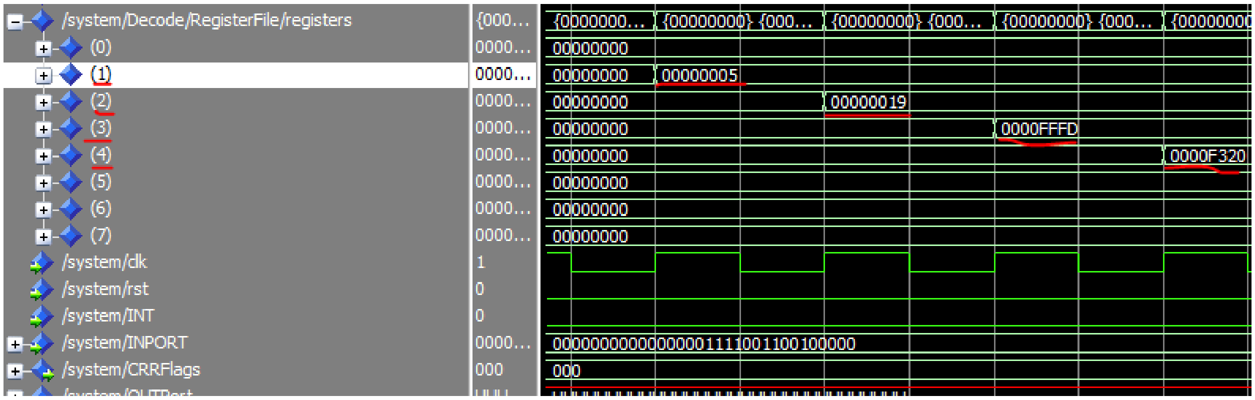
Instruction	**Hazard**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**
IN R1		F	D	E	M	W																	
IN R2			F	D	E	M	W																
IN R3				F	D	E	M	W															
IN R4					F	D	E	M	W														
NOP						**F**	**D**	**E**	**M**	**W**													
IADD R3,R5,2	RAW R3						F	D	E	M	W												
ADD R1,R4,R4								F	D	E	M	W											
NOP									**F**	**D**	**E**	**M**	**W**										
NOP										**F**	**D**	**E**	**M**	**W**									
SUB R5,R4,R6	RAW R5&R4										F	D	E	M	W								
NOP												**F**	**D**	**E**	**M**	**W**							
NOP													**F**	**D**	**E**	**M**	**W**						
AND R7,R6,R6	RAW R6													F	D	E	M	W					
OR R2,R1,R1															F	D	E	M	W				
SHL R2,2																F	D	E	M	W			
NOP																	**F**	**D**	**E**	**M**	**W**		
NOP																		**F**	**D**	**E**	**M**	**W**	
SHR R2,3	RAW R2																		F	D	E	M	W

Instruction	Hazard	19	20	21	22	23	24	25	26	27	28
NOP		F	D	E	M	W					
NOP			F	D	E	M	W				
SWAP R2,R5				F	D	E	M	W			
NOP					F	D	E	M	W		
NOP						F	D	E	M	W	
ADD R5,R2,R2	RAW R2 , R5						F	D	E	M	W

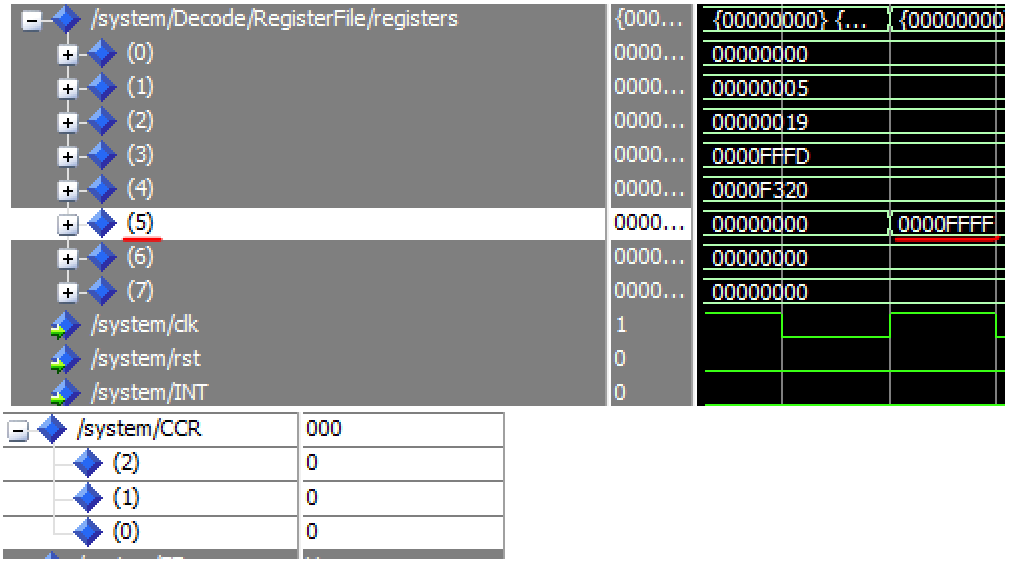
- After adding NOP operations like above table , all instructions gives the correct result
- With forwarding only
  All instructions works CORRECTLY and give the right results with no hazards because all hazards are data dependency so forward unit can handle it

➢ The following wave shows the correct result in register file and CCR flags

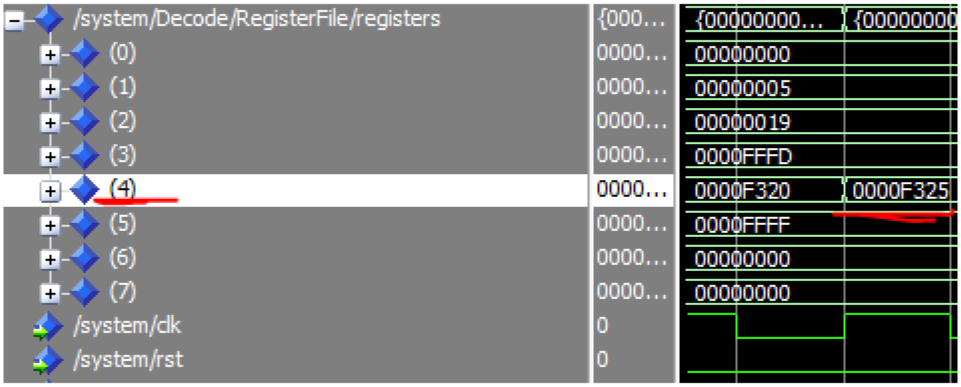
- in R1
- in R2
- in R3
- in R4

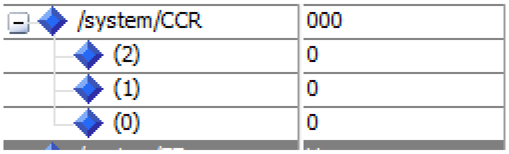

/system/Decode/RegisterFile/registers {000… {0000000… {00000000} {000… {00000000} {000… {00000000} {000… {00000000
(0) 0000… 00000000
(1) 0000… 00000000 00000005
(2) 0000… 00000000 00000019
(3) 0000… 00000000 0000FFFD
(4) 0000… 00000000 0000F320
(5) 0000… 00000000
(6) 0000… 00000000
(7) 0000… 00000000
/system/clk 1
/system/rst 0
/system/INT 0
/system/INPORT 0000… 00000000000000001111001100100000
/system/CRRFlags 000 000
/system/OUTPort

- IADD R3,R5,2



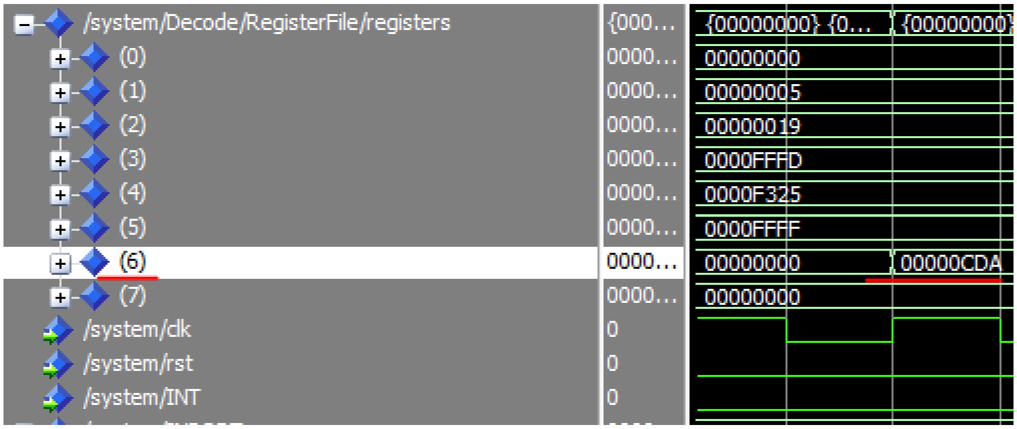
- ADD R1,R4,R4



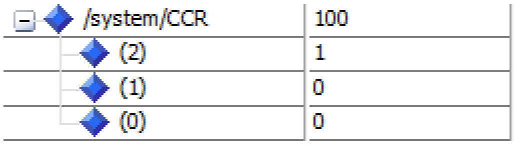


```
/system/CCR   000
    (2)       0
    (1)       0
    (0)       0
```

- SUB R5,R4,R6

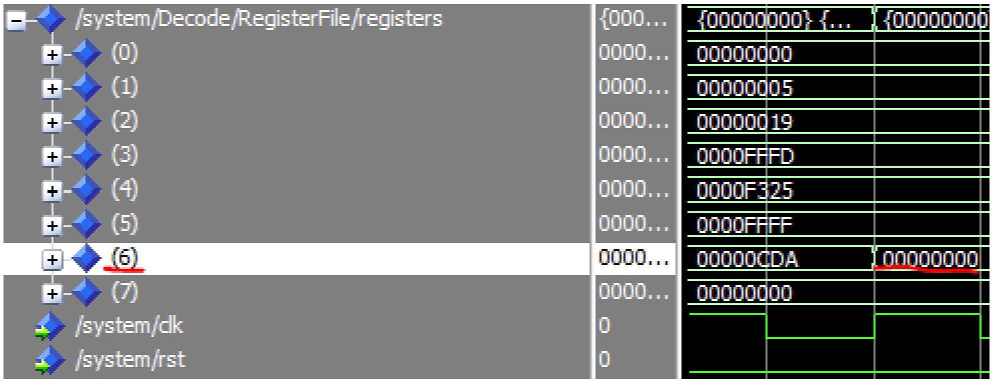


```
/system/Decode/RegisterFile/registers   {000...   {00000000} {0...   {00000000}
    (0)                                 0000...   00000000
    (1)                                 0000...   00000005
    (2)                                 0000...   00000019
    (3)                                 0000...   0000FFFD
    (4)                                 0000...   0000F325
    (5)                                 0000...   0000FFFF
    (6)                                 0000...   00000000           00000CDA
    (7)                                 0000...   00000000
/system/clk                             0
/system/rst                             0
/system/INT                             0
```

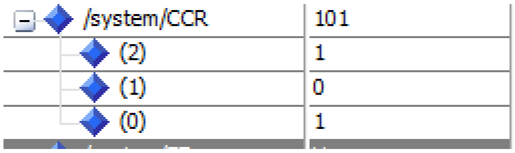


```
/system/CCR   100
    (2)       1
    (1)       0
    (0)       0
```

- AND R7,R6,R6

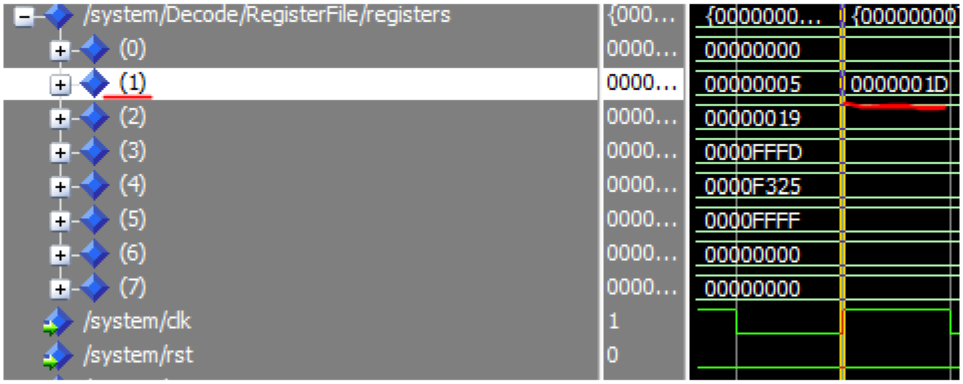


```
/system/Decode/RegisterFile/registers   {000...   {00000000} {...   {00000000
    (0)                                 0000...   00000000
    (1)                                 0000...   00000005
    (2)                                 0000...   00000019
    (3)                                 0000...   0000FFFD
    (4)                                 0000...   0000F325
    (5)                                 0000...   0000FFFF
    (6)                                 0000...   00000CDA          00000000
    (7)                                 0000...   00000000
/system/clk                             0
/system/rst                             0
```

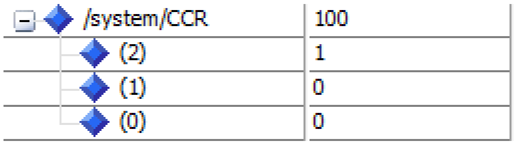


```
/system/CCR   101
    (2)       1
    (1)       0
    (0)       1
```

- OR   R2,R1,R1

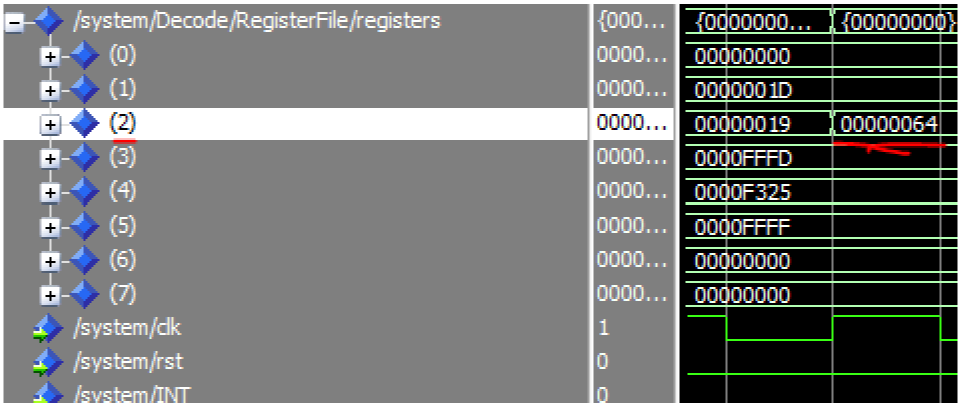


```
/system/Decode/RegisterFile/registers   {000...   {0000000...  {00000000}
  (0)                                   0000...   00000000
  (1)                                   0000...   00000005     0000001D
  (2)                                   0000...   00000019
  (3)                                   0000...   0000FFFD
  (4)                                   0000...   0000F325
  (5)                                   0000...   0000FFFF
  (6)                                   0000...   00000000
  (7)                                   0000...   00000000
/system/clk                             1
/system/rst                             0
```

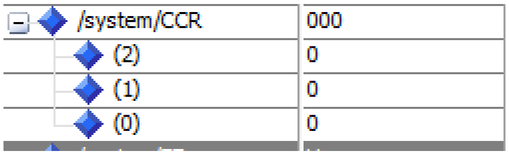


```
/system/CCR   100
  (2)         1
  (1)         0
  (0)         0
```

- SHL R2,2

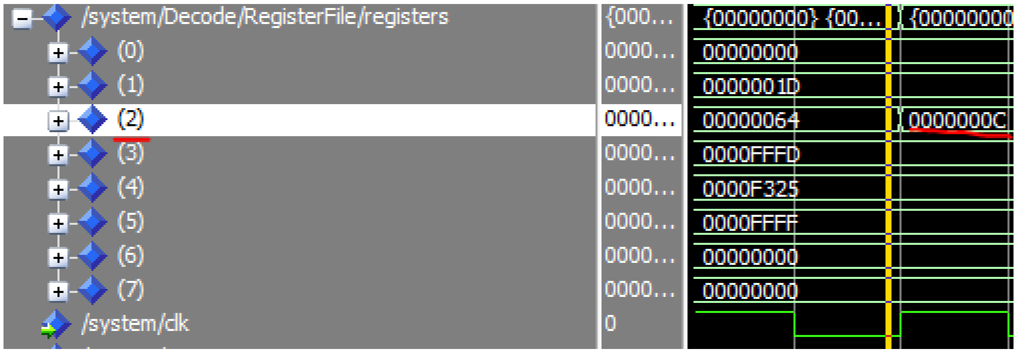


```
/system/Decode/RegisterFile/registers   {000...   {0000000...  {00000000}
  (0)                                   0000...   00000000
  (1)                                   0000...   0000001D
  (2)                                   0000...   00000019     00000064
  (3)                                   0000...   0000FFFD
  (4)                                   0000...   0000F325
  (5)                                   0000...   0000FFFF
  (6)                                   0000...   00000000
  (7)                                   0000...   00000000
/system/clk                             1
/system/rst                             0
/system/INT                             0
```

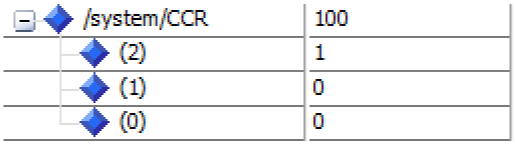


```
/system/CCR   000
  (2)         0
  (1)         0
  (0)         0
```

- SHR R2,3

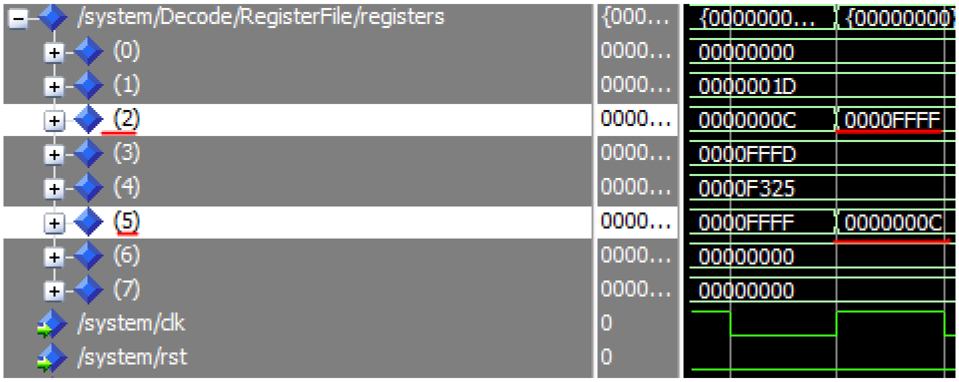


```
/system/Decode/RegisterFile/registers   {000...   {00000000} {00...  {00000000
  (0)                                   0000...   00000000
  (1)                                   0000...   0000001D
  (2)                                   0000...   00000064           0000000C
  (3)                                   0000...   0000FFFD
  (4)                                   0000...   0000F325
  (5)                                   0000...   0000FFFF
  (6)                                   0000...   00000000
  (7)                                   0000...   00000000
/system/clk                             0
```

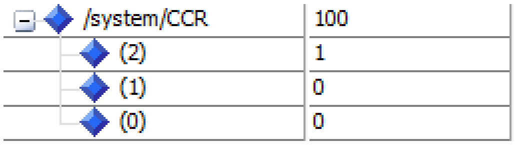


```
/system/CCR   100
  (2)         1
  (1)         0
  (0)         0
```

- SWAP R2,R5

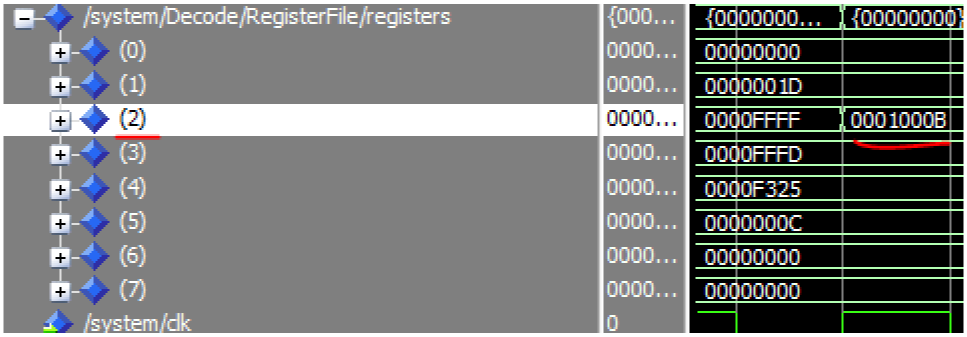


/system/Decode/RegisterFile/registers	{000...	{0000000...	{00000000}
(0)	0000...	00000000	
(1)	0000...	0000001D	
(2)	0000...	0000000C	0000FFFF
(3)	0000...	0000FFFD	
(4)	0000...	0000F325	
(5)	0000...	0000FFFF	0000000C
(6)	0000...	00000000	
(7)	0000...	00000000	
/system/clk	0		
/system/rst	0		

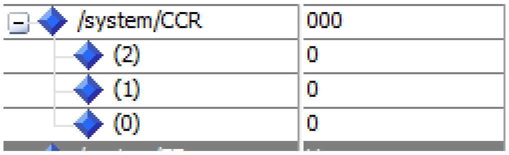


/system/CCR	100
(2)	1
(1)	0
(0)	0

- ADD R5,R2,R2



/system/Decode/RegisterFile/registers	{000...	{0000000...	{00000000}
(0)	0000...	00000000	
(1)	0000...	0000001D	
(2)	0000...	0000FFFF	0001000B
(3)	0000...	0000FFFD	
(4)	0000...	0000F325	
(5)	0000...	0000000C	
(6)	0000...	00000000	
(7)	0000...	00000000	
/system/clk	0		



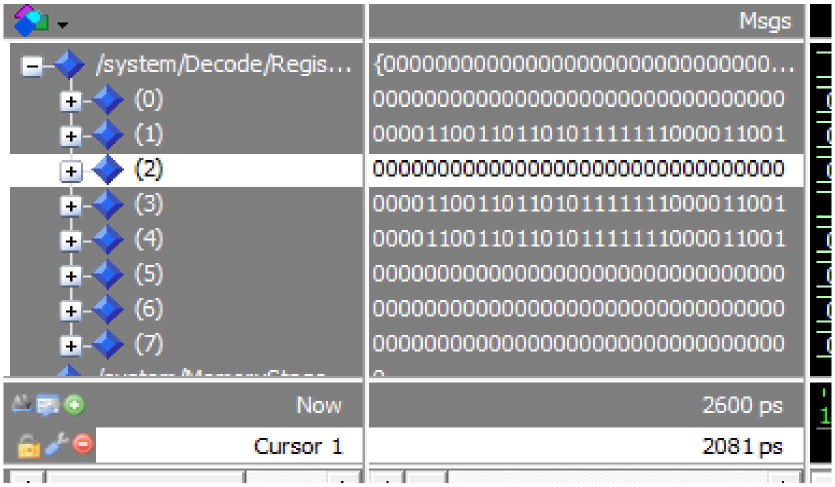
/system/CCR	000
(2)	0
(1)	0
(0)	0

- After adding hazard detection with forward unit give the same result with forward unit only because forward unit can handle all data dependency hazards
- Simlarly , after adding flush with hazard detection unit and forward unit , result are the same with forward unit only .

Memory

Without forwarding unit or hazard detection unit:

Results:



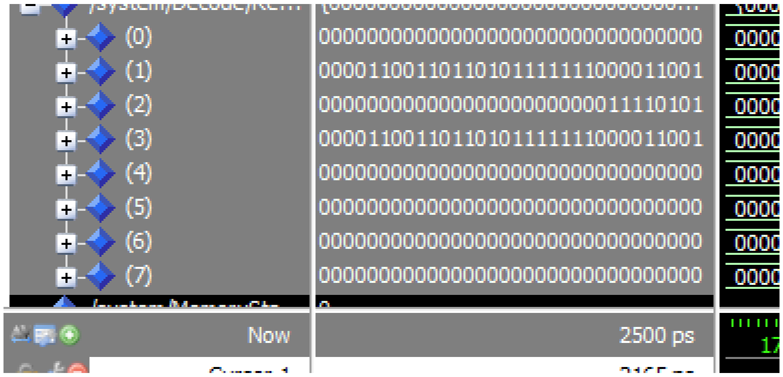
```
                                   Msgs
/system/Decode/Regis...   {0000000000000000000000000000000...
  (0)                     00000000000000000000000000000000
  (1)                     00001100110110101111111000011001
  (2)                     00000000000000000000000000000000
  (3)                     00001100110110101111111000011001
  (4)                     00001100110110101111111000011001
  (5)                     00000000000000000000000000000000
  (6)                     00000000000000000000000000000000
  (7)                     00000000000000000000000000000000

                    Now                           2600 ps
               Cursor 1                           2081 ps
```

instr	hazard	1	2	3	4	5	6	7	8	9	10	11	12	13
In r2		f	d	e	m	w								
In r3			f	d	e	m	w							
In r4				f	d	e	m	w						
Ldm r1,f5					f	d	e	m	w					
nop						f	d	e	m	w				
nop							f	d	e	m	w			

Push r1	Raw Data hazard							f	d	e	m	w		
Push r2									f	d	e	m	w	
Pop r1	WAR									f	d	e	m	w
		10	11	12	13	14	15	16	17	18	19	20	21	22
Pop r2	WAR	f	d	e	m	w								
nop			f	d	e	m	w							
nop				f	d	e	m	w						
Std r2,200	Raw				f	d	e	m	w					
Std r1,202	RaW					f	d	e	m	w				
Ldd r3,202							f	d	e	m	w			
Ldd r4,200								f	d	e	m	w		

With forwarding unit and without hazard detection unit:

Results:

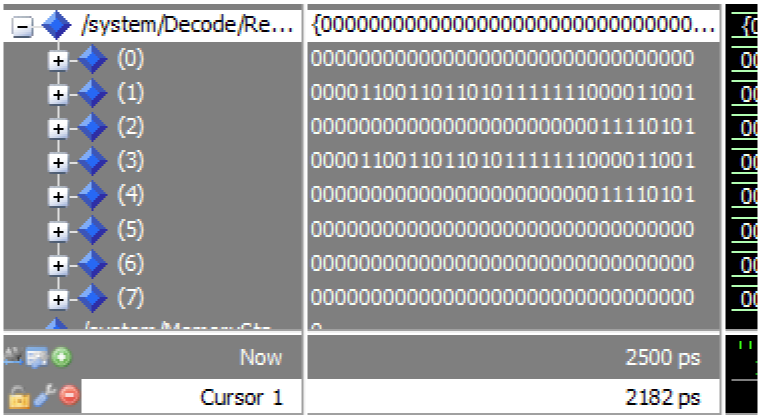


```
(0)  00000000000000000000000000000000   0000
(1)  00001100110110101111111000011001   0000
(2)  00000000000000000000000011110101   0000
(3)  00001100110110101111111000011001   0000
(4)  00000000000000000000000000000000   0000
(5)  00000000000000000000000000000000   0000
(6)  00000000000000000000000000000000   0000
(7)  00000000000000000000000000000000   0000
Now                         2500 ps       17
```

instr	hazard	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
In r2		f	d	e	m	w																
In r3			f	d	e	m	w															
In r4				f	d	e	m	w														
Ldm r1,f5					f	d	e	m	w													
Push r1	Raw Data hazard					f	d	e	m	w												
Push r2							f	d	e	m	w											
Pop r1	WAR							f	d	e	m	w										
Pop r2	WAR								f	d	e	m	w									
nop										f	d	e	m	w								
Std r2,200	Raw										f	d	e	m	w							
Std r1,202	RaW											f	d	e	m	w						
Ldd r3,202													f	d	e	m	w		w			
Ldd r4,200														f	d	e	m	w	m	w		

With forwarding and unit hazard detection unit:

Results:



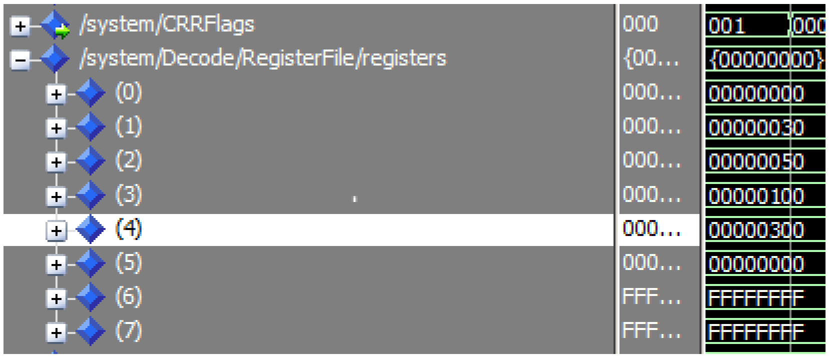
```
/system/Decode/Re...   {00000000000000000000000000000000...
  (0)                  00000000000000000000000000000000
  (1)                  00001100110110101111111000011001
  (2)                  00000000000000000000000011110101
  (3)                  00001100110110101111111000011001
  (4)                  00000000000000000000000011110101
  (5)                  00000000000000000000000000000000
  (6)                  00000000000000000000000000000000
  (7)                  00000000000000000000000000000000
/system/MemorySta...   0
               Now     2500 ps
          Cursor 1     2182 ps
```

Branching

Without forwarding unit & hazard detection unit & flushing

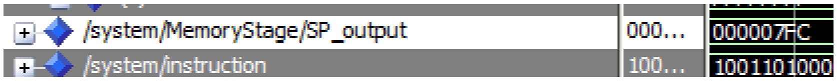
Z flag is CRRFlag(0)
N flag is CRRFlag(1)
C flag is CRRFlag(2)

[1] in R1 ,in R2 ,in R3 ,in R4 ,in R6 ,in R7



```
/system/CRRFlags                         000      001    000
/system/Decode/RegisterFile/registers    {00...   {00000000}
  (0)                                    000...   00000000
  (1)                                    000...   00000030
  (2)                                    000...   00000050
  (3)                                    000...   00000100
  (4)                                    000...   00000300
  (5)                                    000...   00000000
  (6)                                    FFF...   FFFFFFFF
  (7)                                    FFF...   FFFFFFFF
```

[2] Push R4



```
/system/MemoryStage/SP_output    000...   000007FC
/system/instruction              100...   1001101000
```

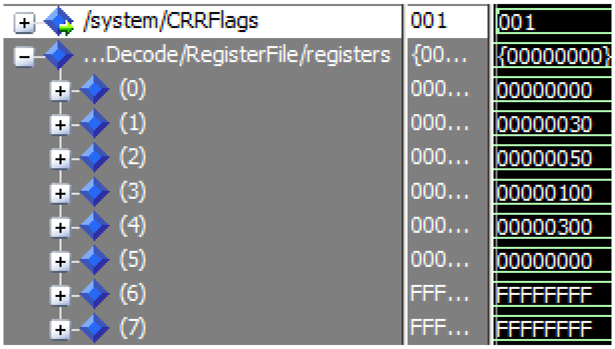
[3] JMP R1

Will jump to 30

[4] INC R7

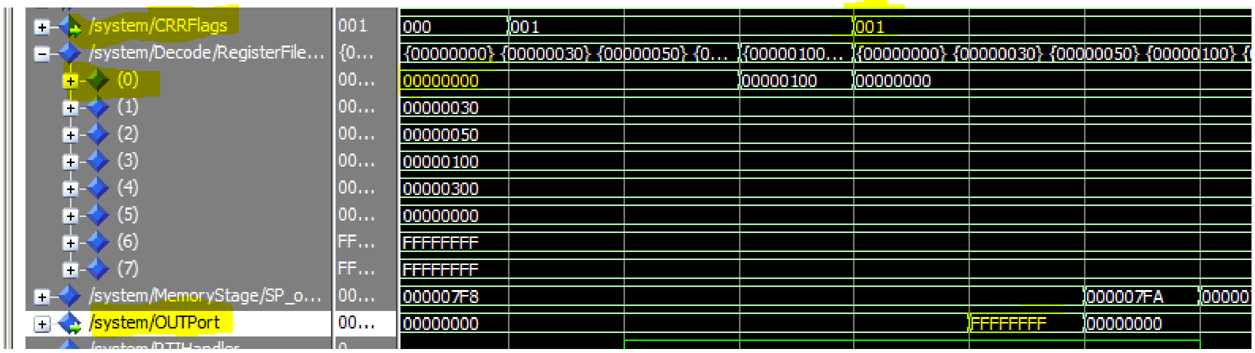
Will not be executed

[5] AND R1,R5,R5 (Z is CRRflgs (0))



```
/system/CRRFlags                   001     001
...Decode/RegisterFile/registers   {00...  {00000000}
  (0)                              000...  00000000
  (1)                              000...  00000030
  (2)                              000...  00000050
  (3)                              000...  00000100
  (4)                              000...  00000300
  (5)                              000...  00000000
  (6)                              FFF...  FFFFFFFF
  (7)                              FFF...  FFFFFFFF
```

[6] ADD R0,R0,R0
out R6



```
/system/CRRFlags                001   000        001                                                      001
/system/Decode/RegisterFile...  {0... {00000000} {00000030} {00000050} {0... {00000100... {00000000} {00000030} {00000050} {00000100} {(
  (0)                           00... 00000000                             00000100      00000000
  (1)                           00... 00000030
  (2)                           00... 00000050
  (3)                           00... 00000100
  (4)                           00... 00000300
  (5)                           00... 00000000
  (6)                           FF... FFFFFFFF
  (7)                           FF... FFFFFFFF
/system/MemoryStage/SP_o...     00... 000007F8                                                                    000007FA    00000
/system/OUTPort                 00... 00000000                                                       FFFFFFFF     00000000
/system/RTIHandler              0
```

An error occurred in the Ram.
The simulation stopped here and didn’t continue

Solve by NOPS

Instruction	Hazard	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
in R1		F	D	E	M	W																	
In R2			F	D	E	M	W																
In R3				F	D	E	M	W															
In R4					F	D	E	M	W														
In R6						F	D	E	M	W													
In R7							F	D	E	M	W												
Push R4								F	D	E	M	W											
Jmp R1									F	D	E	M	W										
AND R1,R5,R5										F	D	E	M	W									

Instruction	Hazards	10	11	12	13	14	15	16	17	18	19	20	21

ADD R0,R0,R0		F	D	E	M	W							
out R6			F	D	E	M	W						
rti				F	D	E	M	W					
JZ R2					F	D	E	M	W				
JZ R3						F	D	E	M	W			
NOT R5							F	D	E	M	W		
NOP								F	D	E	M	W	
NOP									F	D	E	M	W

Instruction	Hazard	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
INC R5	RAW R5 Data Haza	F	D	E	M	W																	
in R6			F	D	E	M	W																
NOP				F	D	E	M	W															
NOP					F	D	E	M	W														

JZ R6	**RAW R6 Data Hazard**					**F**	**D**	**E**	**M**	**W**													
POP R6							**F**	**D**	**E**	**M**	**W**												
Call R6								**F**	**D**	**E**	**M**	**W**											
Add R3,R6,R6									**F**	**D**	**E**	**M**	**W**										
Add R1,R2,R1										**F**	**D**	**E**	**M**	**W**									
ret											**F**	**D**	**E**	**M**	**W**								
INC R6												**F**	**D**	**E**	**M**	**W**							
NOP													**F**	**D**	**E**	**M**	**W**						
NOP														**F**	**D**	**E**	**M**	**W**					

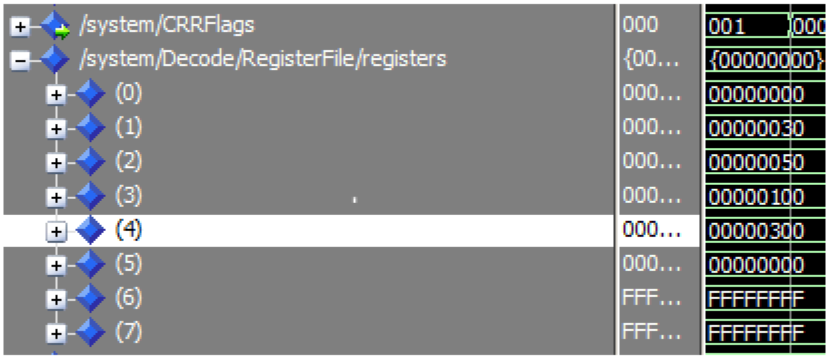
Branching: With Forwarding unit only

Z flag is CRRFlag(0)

N flag is CRRFlag(1)

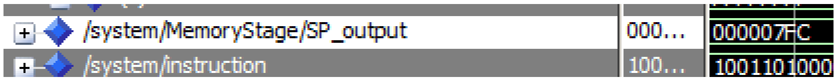
C flag is CRRFlag(2)

[1] in R1 ,in R2 ,in R3 ,in R4 ,in R6 ,in R7



```
/system/CRRFlags                        000    001  000
/system/Decode/RegisterFile/registers   {00...  {00000000}
  (0)                                   000...  00000000
  (1)                                   000...  00000030
  (2)                                   000...  00000050
  (3)                                   000...  00000100
  (4)                                   000...  00000300
  (5)                                   000...  00000000
  (6)                                   FFF...  FFFFFFFF
  (7)                                   FFF...  FFFFFFFF
```

[2] Push R4



```
/system/MemoryStage/SP_output   000...  000007FC
/system/instruction             100...  1001101000
```

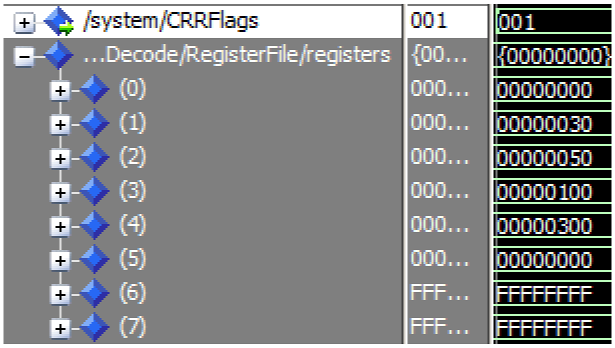
[3] JMP R1

Will jump to 30

[4] INC R7

Will not be executed

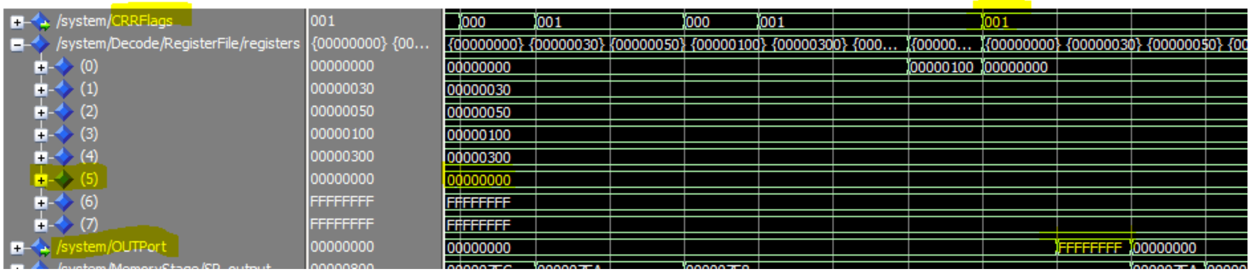
[5] AND R1,R5,R5 (Z is CRRflgs (0))



```
/system/CRRFlags                  001     001
...Decode/RegisterFile/registers  {00...  {00000000}
  (0)                             000...  00000000
  (1)                             000...  00000030
  (2)                             000...  00000050
  (3)                             000...  00000100
  (4)                             000...  00000300
  (5)                             000...  00000000
  (6)                             FFF...  FFFFFFFF
  (7)                             FFF...  FFFFFFFF
```

[6] ADD R0,R0,R0

out R6



```
/system/CRRFlags                   001                  000  001  000  001  001
/system/Decode/RegisterFile/registers {00000000} {00...  {00000000} {00000030} {00000050} {00000100} {00000300} {000... {00000... {00000000} {00000030} {00000050} {00
  (0)                              00000000             00000000  00000100  00000000
  (1)                              00000030             00000030
  (2)                              00000050             00000050
  (3)                              00000100             00000100
  (4)                              00000300             00000300
  (5)                              00000000             00000000
  (6)                              FFFFFFFF             FFFFFFFF
  (7)                              FFFFFFFF             FFFFFFFF
/system/OUTPort                    00000000             00000000  FFFFFFFF  00000000
/system/MemoryStage/SP_output      00000800             000007FC  000007FA  000007F8  000007FA  00000
```

An error occurred in the Ram.

The simulation stopped here and didn’t continue

Branching: Solve by NOPS with forwarding unit

Instruction	Hazard	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
in R1		F	D	E	M	W																	
In R2			F	D	E	M	W																
In R3				F	D	E	M	W															
In R4					F	D	E	M	W														
In R6						F	D	E	M	W													
In R7							F	D	E	M	W												
Push R4								F	D	E	M	W											
Jmp R1									F	D	E	M	W										
AND R1,R5,R5										F	D	E	M	W									

Instruction	Hazards	10	11	12	13	14	15	16	17	18	19	20	21
ADD R0,R0,R0		F	D	E	M	W							

out R6			F	D	E	M	W						
rti				F	D	E	M	W					
JZ R2					F	D	E	M	W				
JZ R3						F	D	E	M	W			
NOT R5							F	D	E	M	W		
INC R5								F	D	E	M	W	
in R6									F	D	E	M	W

Instruction	Hazard	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
NOP		F	D	E	M	W																	
NOP			F	D	E	M	W																
JZ R6	RAW R6 Data Hazard			F	D	E	M	W															
POP R6					F	D	E	M	W														
Call R6						F	D	E	M	W													

Add R3,R6,R6							**F**	**D**	**E**	**M**	**W**												
Add R1,R2,R1								**F**	**D**	**E**	**M**	**W**											
ret									**F**	**D**	**E**	**M**	**W**										
INC R6										**F**	**D**	**E**	**M**	**W**									
NOP											**F**	**D**	**E**	**M**	**W**								
NOP												**F**	**D**	**E**	**M**	**W**							

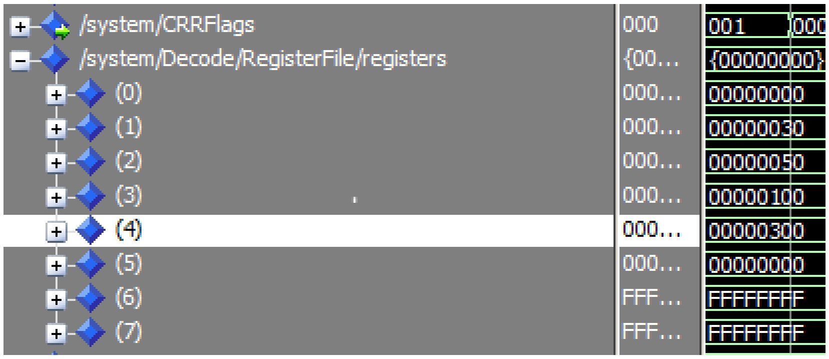
Branching: With Forwarding unit & Hazard detection unit only

Z flag is CRRFlag(0)

N flag is CRRFlag(1)

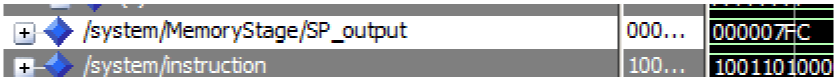
C flag is CRRFlag(2)

[1] in R1 ,in R2 ,in R3 ,in R4 ,in R6 ,in R7



```
/system/CRRFlags                          000      001  000
/system/Decode/RegisterFile/registers     {00...   {00000000}
  (0)                                     000...   00000000
  (1)                                     000...   00000030
  (2)                                     000...   00000050
  (3)                                     000...   00000100
  (4)                                     000...   00000300
  (5)                                     000...   00000000
  (6)                                     FFF...   FFFFFFFF
  (7)                                     FFF...   FFFFFFFF
```

[2] Push R4



```
/system/MemoryStage/SP_output     000...   000007FC
/system/instruction               100...   1001101000
```

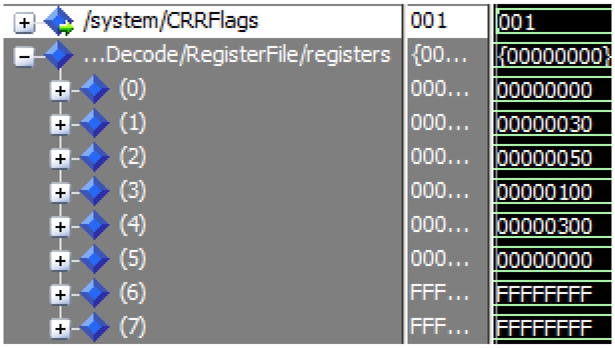
[3] JMP R1

Will jump to 30

[4] INC R7

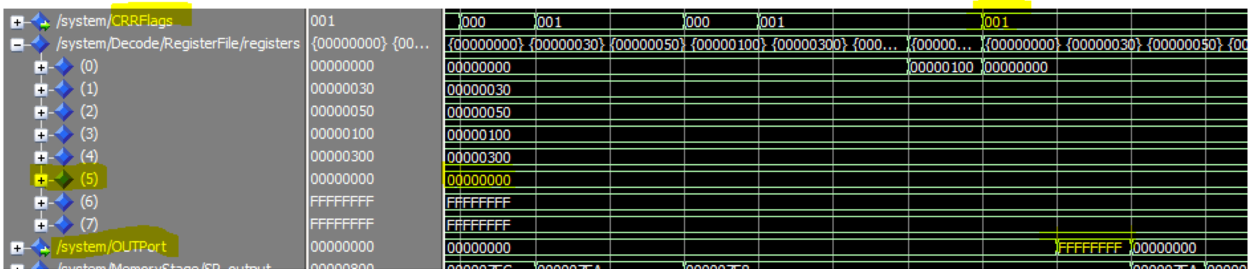
Will not be executed

[5] AND R1,R5,R5 (Z is CRRflgs (0))



```
/system/CRRFlags                 001        001
...Decode/RegisterFile/registers {00...     {00000000}
  (0)                            000...     00000000
  (1)                            000...     00000030
  (2)                            000...     00000050
  (3)                            000...     00000100
  (4)                            000...     00000300
  (5)                            000...     00000000
  (6)                            FFF...     FFFFFFFF
  (7)                            FFF...     FFFFFFFF
```

[6] ADD R0,R0,R0
out R6

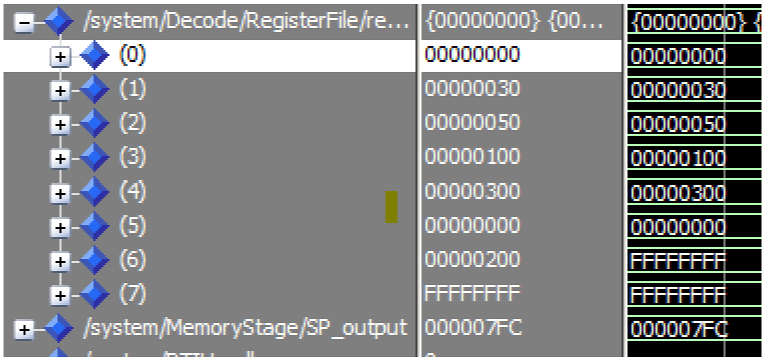


```
/system/CRRFlags                          001                  000  001  000  001  001
/system/Decode/RegisterFile/registers     {00000000} {00...    {00000000} {00000030} {00000050} {00000100} {00000300} {000... {00000... {00000000} {00000030} {00000050} {00
  (0)                                     00000000             00000000  00000100  00000000
  (1)                                     00000030             00000030
  (2)                                     00000050             00000050
  (3)                                     00000100             00000100
  (4)                                     00000300             00000300
  (5)                                     00000000             00000000
  (6)                                     FFFFFFFF             FFFFFFFF
  (7)                                     FFFFFFFF             FFFFFFFF
/system/OUTPort                           00000000             00000000  FFFFFFFF  00000000
```

An error occurred in the Ram.
The simulation stopped here and didn’t continue

Branching: With forwarding & hazard & flushing units

[1] in R1 , in R2 , in R3 ,   in R4 ,   in R6   ,in R7,   Push R4

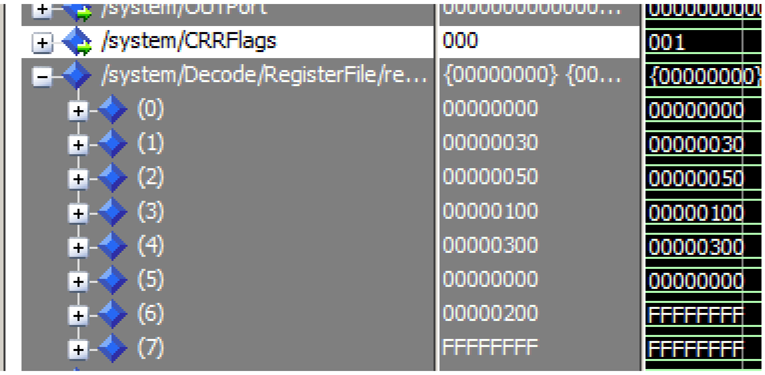


```
/system/Decode/RegisterFile/re...  {00000000} {00...  {00000000} {
  (0)                              00000000           00000000
  (1)                              00000030           00000030
  (2)                              00000050           00000050
  (3)                              00000100           00000100
  (4)                              00000300           00000300
  (5)                              00000000           00000000
  (6)                              00000200           FFFFFFFF
  (7)                              FFFFFFFF           FFFFFFFF
/system/MemoryStage/SP_output      000007FC           000007FC
```

[2] JMP R1
Will jump to 30

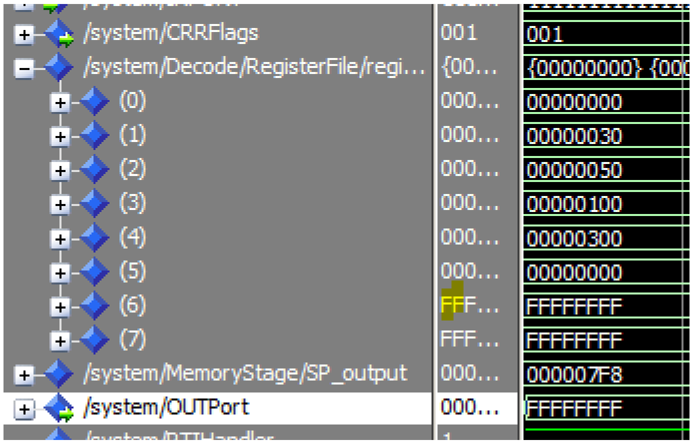
[3] INC R7
this statement will not be executed

[4] AND R1,R5,R5   (Z flag is CRRFlag(0))



```
/system/OUTPort                    0000000000000...   000000000
/system/CRRFlags                   000                001
/system/Decode/RegisterFile/re...  {00000000} {00...  {00000000}
  (0)                              00000000           00000000
  (1)                              00000030           00000030
  (2)                              00000050           00000050
  (3)                              00000100           00000100
  (4)                              00000300           00000300
  (5)                              00000000           00000000
  (6)                              00000200           FFFFFFFF
  (7)                              FFFFFFFF           FFFFFFFF
```

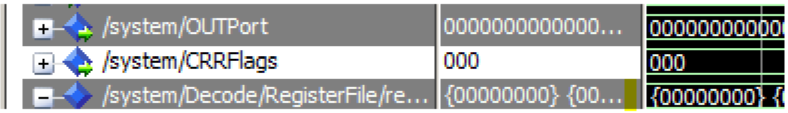
[5] ADD R0,R0,R0 (Z flag is CRRFlag(0))
Out R6



```
/system/CRRFlags                        001         001
/system/Decode/RegisterFile/regi...     {00...      {00000000} {000
  (0)                                   000...      00000000
  (1)                                   000...      00000030
  (2)                                   000...      00000050
  (3)                                   000...      00000100
  (4)                                   000...      00000300
  (5)                                   000...      00000000
  (6)                                   FFF...      FFFFFFFF
  (7)                                   FFF...      FFFFFFFF
/system/MemoryStage/SP_output           000...      000007F8
/system/OUTPort                         000...      FFFFFFFF
```

[6] JZ R2   (Z flag is CRRFlag(0))

Will Jump to 50



```
/system/OUTPort                         0000000000000...    000000000000
/system/CRRFlags                        000                 000
/system/Decode/RegisterFile/re...       {00000000} {00...   {00000000} {
```

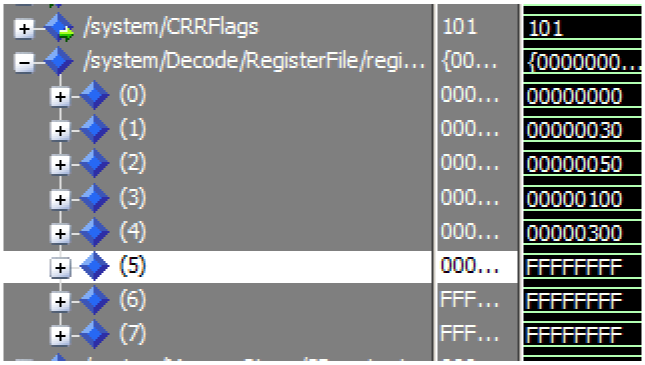
[7] INC R7

This statement will not be executed

[8] JZ R3

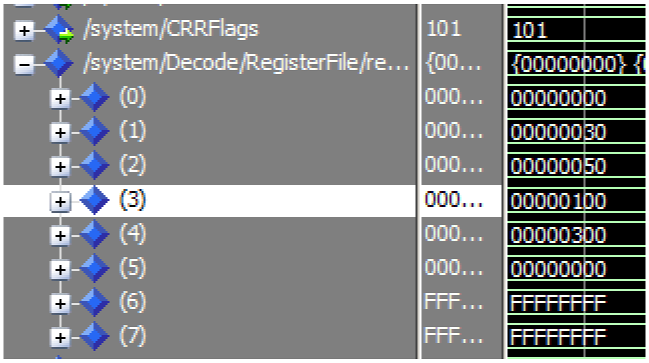
Jump Not taken

[9] NOT R5



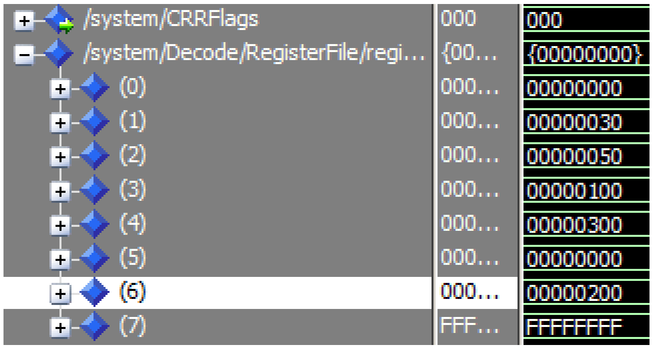
```
/system/CRRFlags                        101         101
/system/Decode/RegisterFile/regi...     {00...      {0000000...
  (0)                                   000...      00000000
  (1)                                   000...      00000030
  (2)                                   000...      00000050
  (3)                                   000...      00000100
  (4)                                   000...      00000300
  (5)                                   000...      FFFFFFFF
  (6)                                   FFF...      FFFFFFFF
  (7)                                   FFF...      FFFFFFFF
```

[10] INC R5 #R5=0, Z=1, C=1, N=0



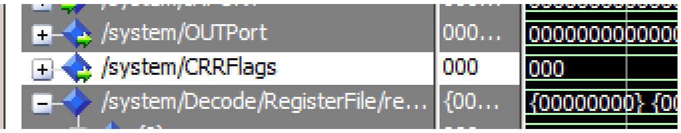
```
/system/CRRFlags                    101     101
/system/Decode/RegisterFile/re...   {00...  {00000000} {(
  (0)                               000...  00000000
  (1)                               000...  00000030
  (2)                               000...  00000050
  (3)                               000...  00000100
  (4)                               000...  00000300
  (5)                               000...  00000000
  (6)                               FFF...  FFFFFFFF
  (7)                               FFF...  FFFFFFFF
```

[11] in R6 #R6=200, flag no change



```
/system/CRRFlags                    000     000
/system/Decode/RegisterFile/regi... {00...  {00000000}
  (0)                               000...  00000000
  (1)                               000...  00000030
  (2)                               000...  00000050
  (3)                               000...  00000100
  (4)                               000...  00000300
  (5)                               000...  00000000
  (6)                               000...  00000200
  (7)                               FFF...  FFFFFFFF
```

[11] JZ R6 #jump taken, Z = 0

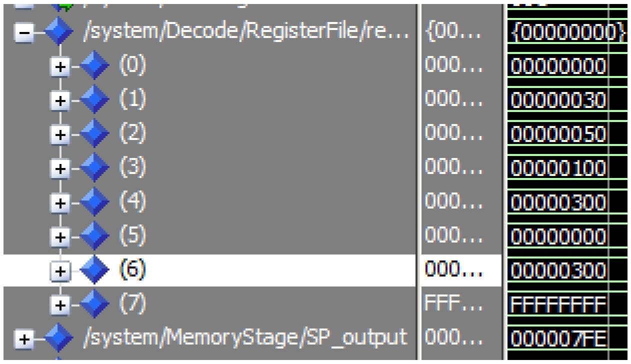


```
/system/OUTPort                     000...  0000000000000(
/system/CRRFlags                    000     000
/system/Decode/RegisterFile/re...   {00...  {00000000} {00
```

[12] INC R1

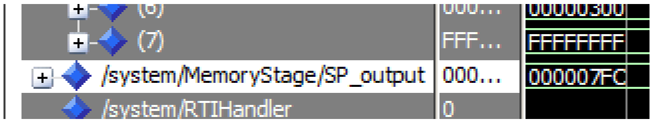
This statement will not be executed

[13] POP R6 #R6=300, SP=7FE



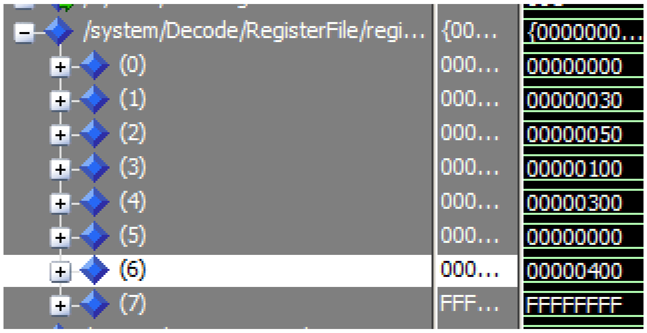
```
/system/Decode/RegisterFile/re...   {00...   {00000000}
   (0)                              000...   00000000
   (1)                              000...   00000030
   (2)                              000...   00000050
   (3)                              000...   00000100
   (4)                              000...   00000300
   (5)                              000...   00000000
   (6)                              000...   00000300
   (7)                              FFF...   FFFFFFFF
/system/MemoryStage/SP_output       000...   000007FE
```

[14] Call R6



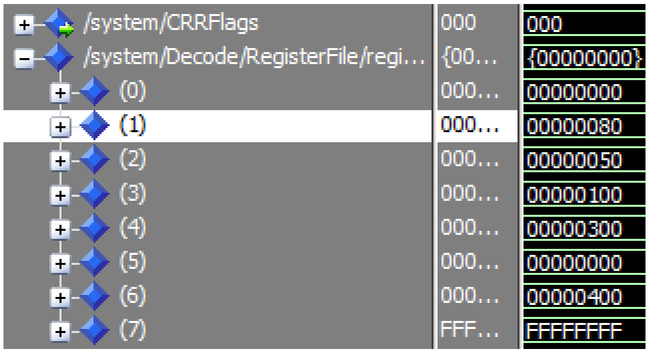
```
   (6)                              000...   00000300
   (7)                              FFF...   FFFFFFFF
/system/MemoryStage/SP_output       000...   000007FC
/system/RTIHandler                  0
```

[15] Add R3,R6,R6 #R6=400



```
/system/Decode/RegisterFile/regi... {00...   {0000000...
   (0)                              000...   00000000
   (1)                              000...   00000030
   (2)                              000...   00000050
   (3)                              000...   00000100
   (4)                              000...   00000300
   (5)                              000...   00000000
   (6)                              000...   00000400
   (7)                              FFF...   FFFFFFFF
```

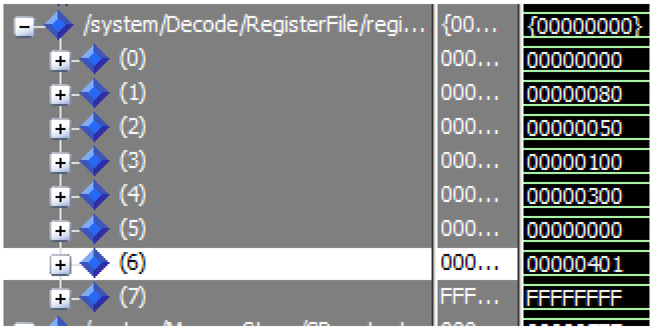
[16] Add R1,R2,R1 #R1=80, C->0,N=0, Z=0



```
/system/CRRFlags                    000      000
/system/Decode/RegisterFile/regi... {00...   {00000000}
   (0)                              000...   00000000
   (1)                              000...   00000080
   (2)                              000...   00000050
   (3)                              000...   00000100
   (4)                              000...   00000300
   (5)                              000...   00000000
   (6)                              000...   00000400
   (7)                              FFF...   FFFFFFFF
```

[17] ret

[18] INC R6

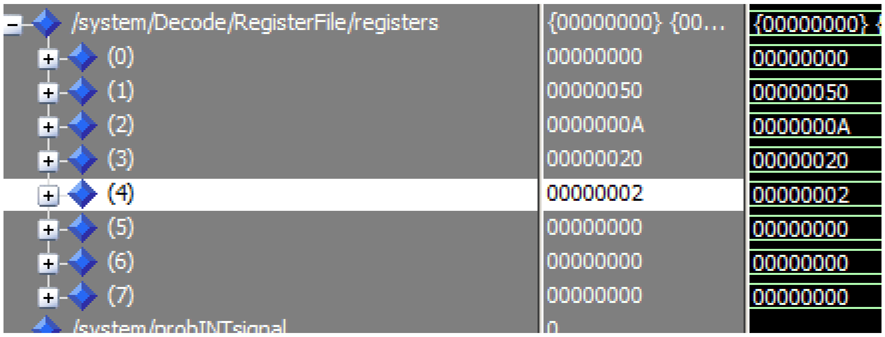


```
/system/Decode/RegisterFile/regi...  {00...  {00000000}
  (0)                                000...  00000000
  (1)                                000...  00000080
  (2)                                000...  00000050
  (3)                                000...  00000100
  (4)                                000...  00000300
  (5)                                000...  00000000
  (6)                                000...  00000401
  (7)                                FFF...  FFFFFFFF
```

Branching Prediction

Without forwarding & hazard & flushing units

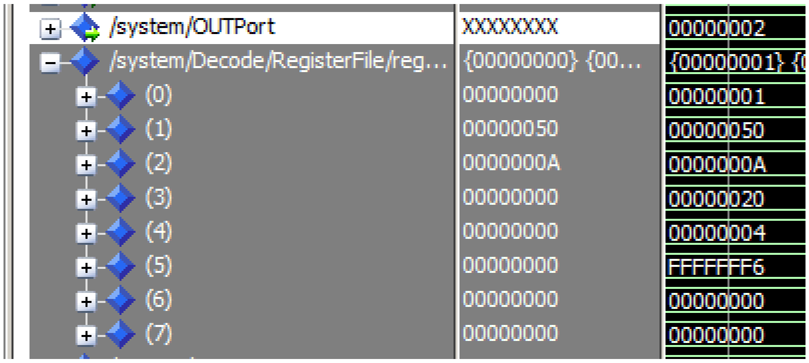
[1] LDM R2,0A
[2] LDM R0,0
[3] LDM R1,50
[4] LDM R3,20
[5] LDM R4,2



```
/system/Decode/RegisterFile/registers   {00000000} {00...   {00000000} {
   (0)                                  00000000            00000000
   (1)                                  00000050            00000050
   (2)                                  0000000A            0000000A
   (3)                                  00000020            00000020
   (4)                                  00000002            00000002
   (5)                                  00000000            00000000
   (6)                                  00000000            00000000
   (7)                                  00000000            00000000
/system/probINTsignal                   0
```

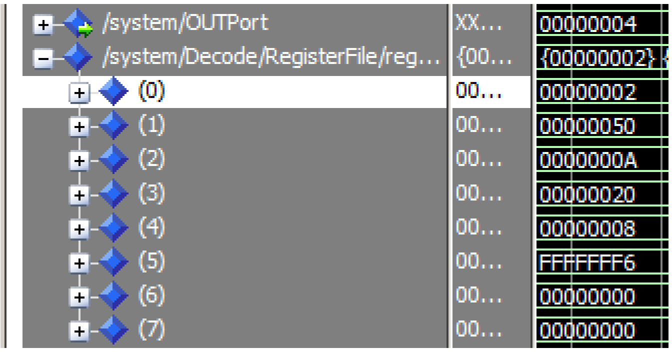
[6]JMP R3
Jump to 20

SUB R0,R2,R5   **(1st iter.)**, JZ R1 , ADD R4,R4,R4 , OUT R4, INC R0, JMP R3



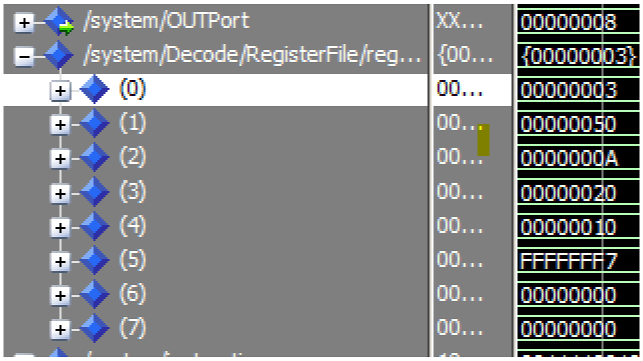
```
/system/OUTPort                        XXXXXXXX            00000002
/system/Decode/RegisterFile/reg...     {00000000} {00...   {00000001} {(
   (0)                                 00000000            00000001
   (1)                                 00000050            00000050
   (2)                                 0000000A            0000000A
   (3)                                 00000000            00000020
   (4)                                 00000000            00000004
   (5)                                 00000000            FFFFFFF6
   (6)                                 00000000            00000000
   (7)                                 00000000            00000000
```

SUB R0,R2,R5   **(2nd iter.)**, JZ R1 , ADD R4,R4,R4 , OUT R4, INC R0, JMP R3



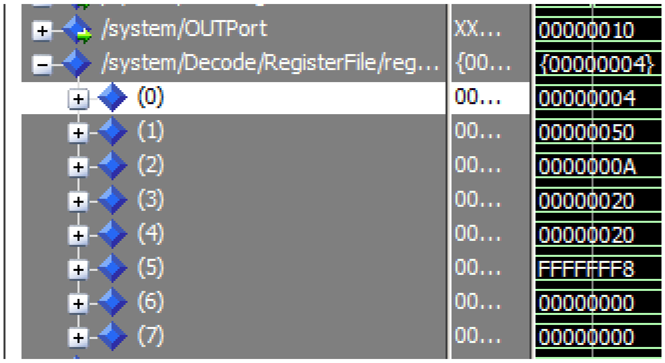
```
/system/OUTPort                     XX...  00000004
/system/Decode/RegisterFile/reg...  {00... {00000002}
  (0)                               00...  00000002
  (1)                               00...  00000050
  (2)                               00...  0000000A
  (3)                               00...  00000020
  (4)                               00...  00000008
  (5)                               00...  FFFFFFF6
  (6)                               00...  00000000
  (7)                               00...  00000000
```

SUB R0,R2,R5 **(3rd iter.)**, JZ R1 , ADD R4,R4,R4 , OUT R4, INC R0, JMP R3



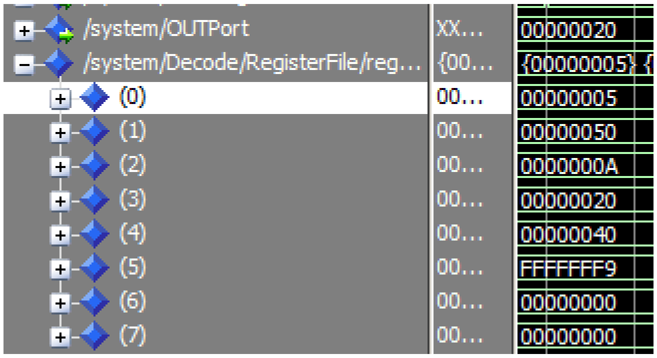
```
/system/OUTPort                     XX...  00000008
/system/Decode/RegisterFile/reg...  {00... {00000003}
  (0)                               00...  00000003
  (1)                               00...  00000050
  (2)                               00...  0000000A
  (3)                               00...  00000020
  (4)                               00...  00000010
  (5)                               00...  FFFFFFF7
  (6)                               00...  00000000
  (7)                               00...  00000000
```

SUB R0,R2,R5 **(4th iter.)**, JZ R1 , ADD R4,R4,R4 , OUT R4, INC R0, JMP R3



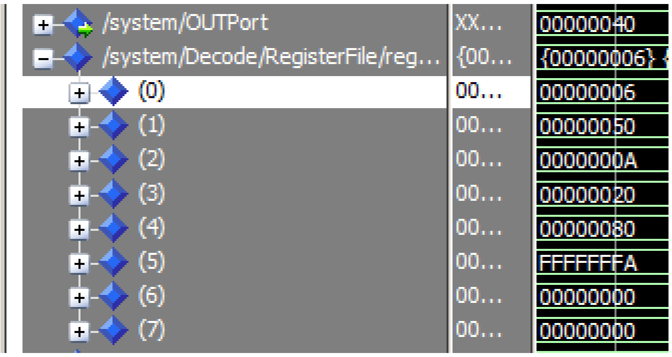
```
/system/OUTPort                     XX...  00000010
/system/Decode/RegisterFile/reg...  {00... {00000004}
  (0)                               00...  00000004
  (1)                               00...  00000050
  (2)                               00...  0000000A
  (3)                               00...  00000020
  (4)                               00...  00000020
  (5)                               00...  FFFFFFF8
  (6)                               00...  00000000
  (7)                               00...  00000000
```

SUB R0,R2,R5 **(5th iter.)**, JZ R1 , ADD R4,R4,R4 , OUT R4, INC R0, JMP R3



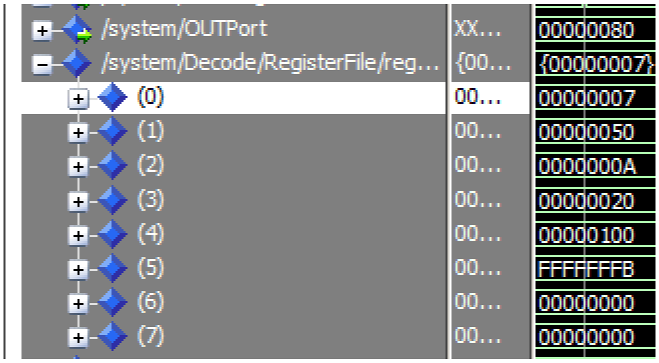
```
/system/OUTPort                      XX...   00000020
/system/Decode/RegisterFile/reg...   {00...  {00000005} {
  (0)                                00...   00000005
  (1)                                00...   00000050
  (2)                                00...   0000000A
  (3)                                00...   00000020
  (4)                                00...   00000040
  (5)                                00...   FFFFFFF9
  (6)                                00...   00000000
  (7)                                00...   00000000
```

SUB R0,R2,R5 **(6th iter.)**, JZ R1 , ADD R4,R4,R4 , OUT R4, INC R0, JMP R3



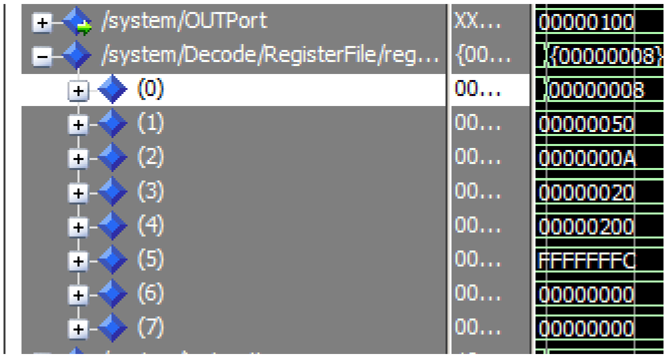
```
/system/OUTPort                      XX...   00000040
/system/Decode/RegisterFile/reg...   {00...  {00000006} {
  (0)                                00...   00000006
  (1)                                00...   00000050
  (2)                                00...   0000000A
  (3)                                00...   00000020
  (4)                                00...   00000080
  (5)                                00...   FFFFFFFA
  (6)                                00...   00000000
  (7)                                00...   00000000
```

SUB R0,R2,R5 **(7th iter.)**, JZ R1 , ADD R4,R4,R4 , OUT R4, INC R0, JMP R3



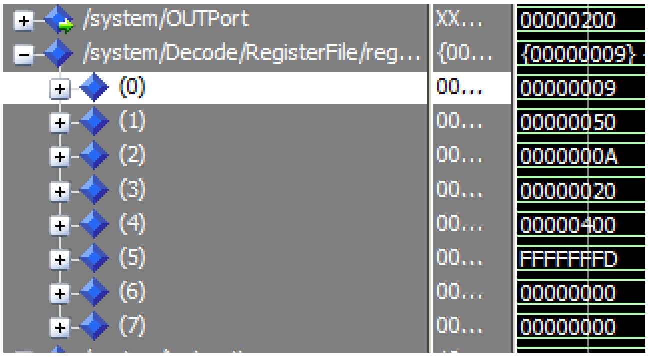
```
/system/OUTPort                      XX...   00000080
/system/Decode/RegisterFile/reg...   {00...  {00000007}
  (0)                                00...   00000007
  (1)                                00...   00000050
  (2)                                00...   0000000A
  (3)                                00...   00000020
  (4)                                00...   00000100
  (5)                                00...   FFFFFFFB
  (6)                                00...   00000000
  (7)                                00...   00000000
```

SUB R0,R2,R5 **(8th iter.)**, JZ R1 , ADD R4,R4,R4 , OUT R4, INC R0, JMP R3



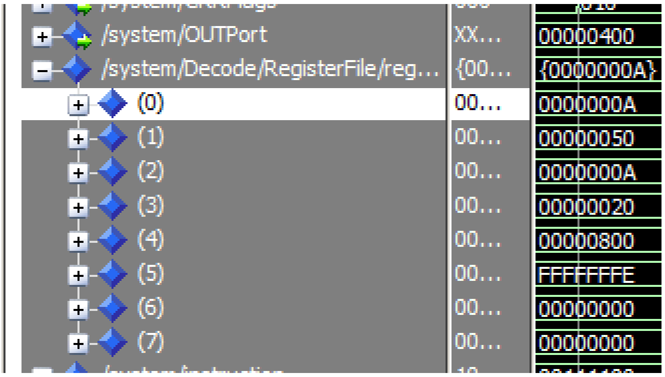
```
/system/OUTPort                     XX...   00000100
/system/Decode/RegisterFile/reg...  {00...  {00000008}
  (0)                               00...   00000008
  (1)                               00...   00000050
  (2)                               00...   0000000A
  (3)                               00...   00000020
  (4)                               00...   00000200
  (5)                               00...   FFFFFFFC
  (6)                               00...   00000000
  (7)                               00...   00000000
```

SUB R0,R2,R5 **(9th iter.)**, JZ R1 , ADD R4,R4,R4 , OUT R4, INC R0, JMP R3



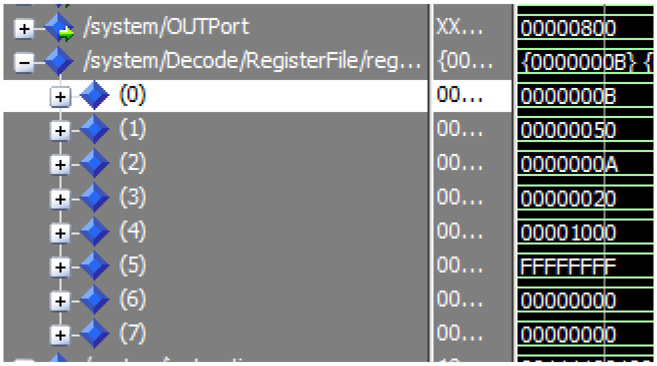
```
/system/OUTPort                     XX...   00000200
/system/Decode/RegisterFile/reg...  {00...  {00000009}
  (0)                               00...   00000009
  (1)                               00...   00000050
  (2)                               00...   0000000A
  (3)                               00...   00000020
  (4)                               00...   00000400
  (5)                               00...   FFFFFFFD
  (6)                               00...   00000000
  (7)                               00...   00000000
```

SUB R0,R2,R5 **(10th iter.)**, JZ R1 , ADD R4,R4,R4 , OUT R4, INC R0, JMP R3



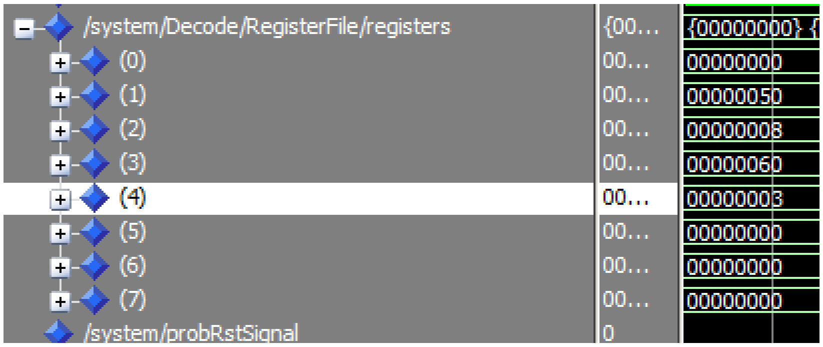
```
/system/OUTPort                     XX...   00000400
/system/Decode/RegisterFile/reg...  {00...  {0000000A}
  (0)                               00...   0000000A
  (1)                               00...   00000050
  (2)                               00...   0000000A
  (3)                               00...   00000020
  (4)                               00...   00000800
  (5)                               00...   FFFFFFFE
  (6)                               00...   00000000
  (7)                               00...   00000000
```

SUB R0,R2,R5 **(11th iter.)**, JZ R1 , ADD R4,R4,R4 , OUT R4, INC R0, JMP R3



```
/system/OUTPort                         XX...  00000800
/system/Decode/RegisterFile/reg...      {00... {0000000B} {
  (0)                                   00...  0000000B
  (1)                                   00...  00000050
  (2)                                   00...  0000000A
  (3)                                   00...  00000020
  (4)                                   00...  00001000
  (5)                                   00...  FFFFFFFF
  (6)                                   00...  00000000
  (7)                                   00...  00000000
```

LDM R0,0 , LDM R2,8 , LDM R3,60 , LDM R4,3

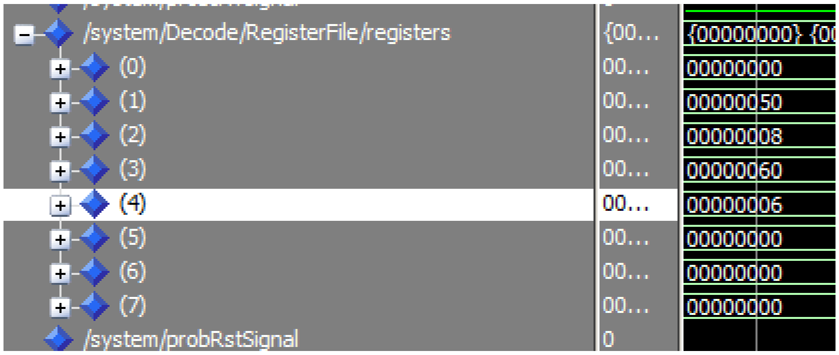


```
/system/Decode/RegisterFile/registers   {00... {00000000} {
  (0)                                   00...  00000000
  (1)                                   00...  00000050
  (2)                                   00...  00000008
  (3)                                   00...  00000060
  (4)                                   00...  00000003
  (5)                                   00...  00000000
  (6)                                   00...  00000000
  (7)                                   00...  00000000
/system/probRstSignal                   0
```

JMP R3

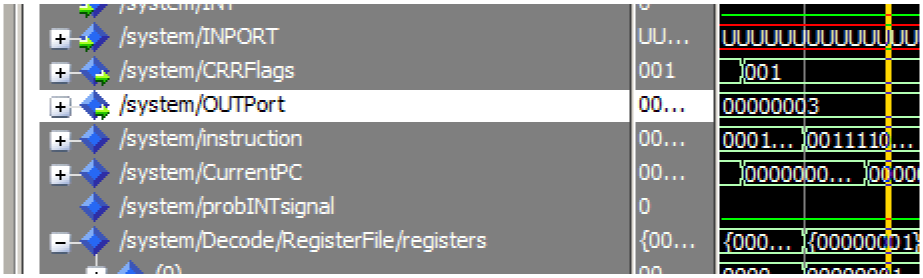
Jump to 60

ADD R4,R4,R4 (1st iter.)



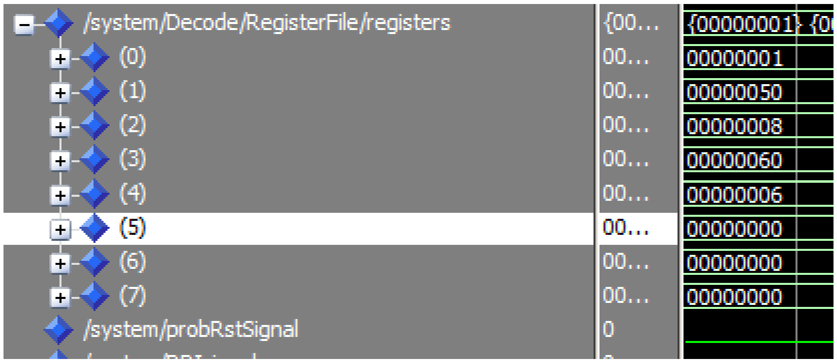
```
/system/Decode/RegisterFile/registers   {00... {00000000} {00
  (0)                                   00...  00000000
  (1)                                   00...  00000050
  (2)                                   00...  00000008
  (3)                                   00...  00000060
  (4)                                   00...  00000006
  (5)                                   00...  00000000
  (6)                                   00...  00000000
  (7)                                   00...  00000000
/system/probRstSignal                   0
```

OUT R4



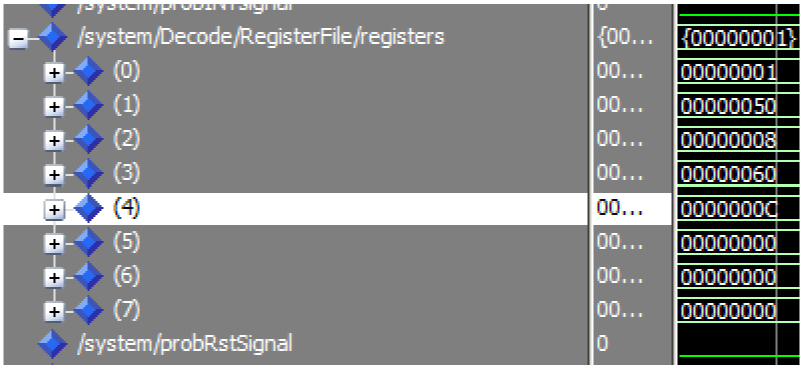
```
/system/INT                              0
/system/INPORT                           UU...   UUUUUUUUUUUUUU
/system/CRRFlags                         001       001
/system/OUTPort                          00...   00000003
/system/instruction                      00...   0001...  0011110...
/system/CurrentPC                        00...     0000000...  0000
/system/probINTsignal                    0
/system/Decode/RegisterFile/registers    {00...  {000...  {00000001}
  (0)                                    00...   0000     00000001
```

INC R0, AND R0,R2,R5 , JZ R3



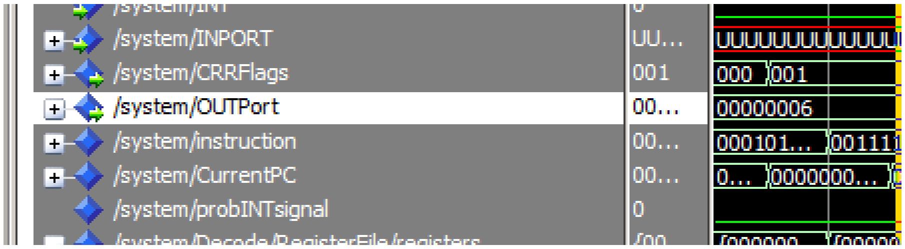
```
/system/Decode/RegisterFile/registers    {00...  {00000001} {0
  (0)                                    00...   00000001
  (1)                                    00...   00000050
  (2)                                    00...   00000008
  (3)                                    00...   00000060
  (4)                                    00...   00000006
  (5)                                    00...   00000000
  (6)                                    00...   00000000
  (7)                                    00...   00000000
/system/probRstSignal                    0
```

ADD R4,R4,R4 (2nd iter.)



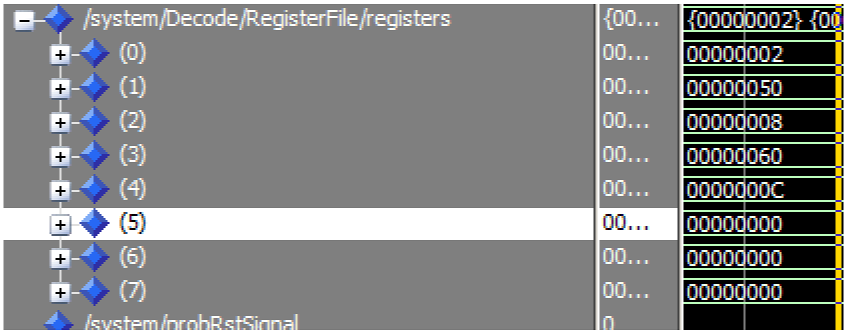
```
/system/probINTsignal                    0
/system/Decode/RegisterFile/registers    {00...  {00000001}
  (0)                                    00...   00000001
  (1)                                    00...   00000050
  (2)                                    00...   00000008
  (3)                                    00...   00000060
  (4)                                    00...   0000000C
  (5)                                    00...   00000000
  (6)                                    00...   00000000
  (7)                                    00...   00000000
/system/probRstSignal                    0
```

OUT R4



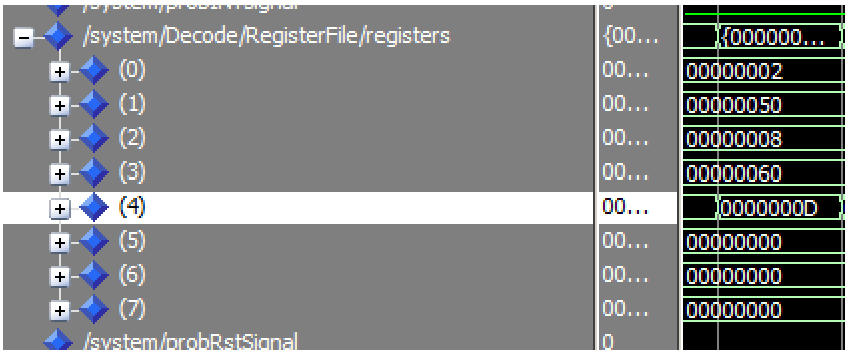
```
/system/INT                              0
/system/INPORT                           UU...   UUUUUUUUUUUUUU
/system/CRRFlags                         001     000   001
/system/OUTPort                          00...   00000006
/system/instruction                      00...   000101...   001111
/system/CurrentPC                        00...   0...  0000000...
/system/probINTsignal                    0
/system/Decode/RegisterFile/registers    {00...  {000000...  {00000
```

INC R0, AND R0,R2,R5 JZ R3



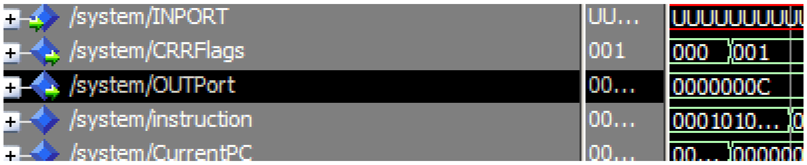
```
/system/Decode/RegisterFile/registers   {00...   {00000002} {00
  (0)                                   00...    00000002
  (1)                                   00...    00000050
  (2)                                   00...    00000008
  (3)                                   00...    00000060
  (4)                                   00...    0000000C
  (5)                                   00...    00000000
  (6)                                   00...    00000000
  (7)                                   00...    00000000
/system/probRstSignal                   0
```

ADD R4,R4,R4 (3rd iteration)



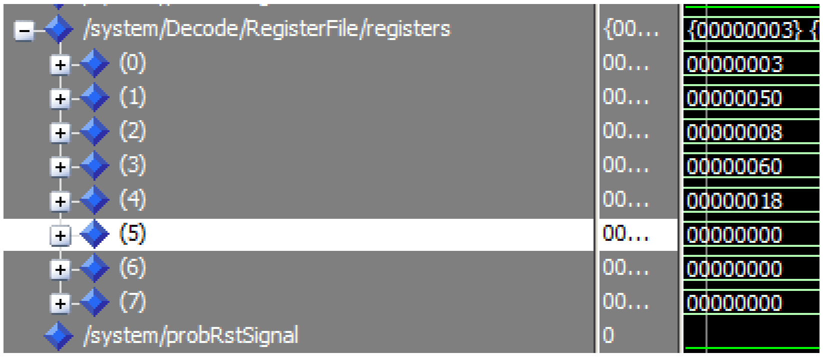
```
/system/probINTSignal                   0
/system/Decode/RegisterFile/registers   {00...   {000000...
  (0)                                   00...    00000002
  (1)                                   00...    00000050
  (2)                                   00...    00000008
  (3)                                   00...    00000060
  (4)                                   00...    0000000D
  (5)                                   00...    00000000
  (6)                                   00...    00000000
  (7)                                   00...    00000000
/system/probRstSignal                   0
```

OUT R4



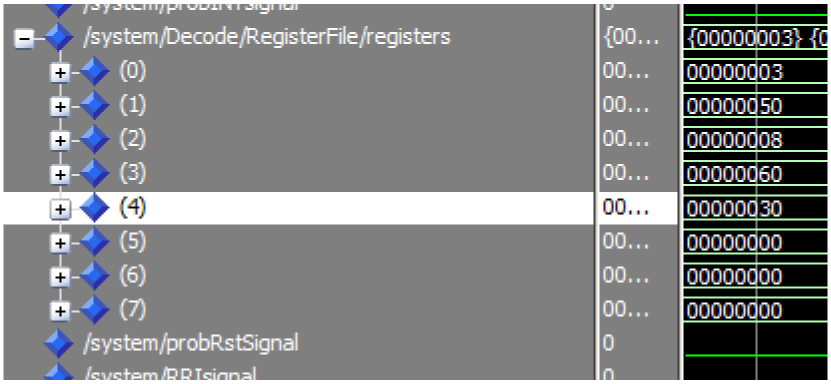
```
/system/INPORT        UU...   UUUUUUUUUU
/system/CRRFlags      001     000   001
/system/OUTPort       00...   0000000C
/system/instruction   00...   0001010...  0
/system/CurrentPC     00...   00...  000000
```

INC R0, AND R0,R2,R5, JZ R3



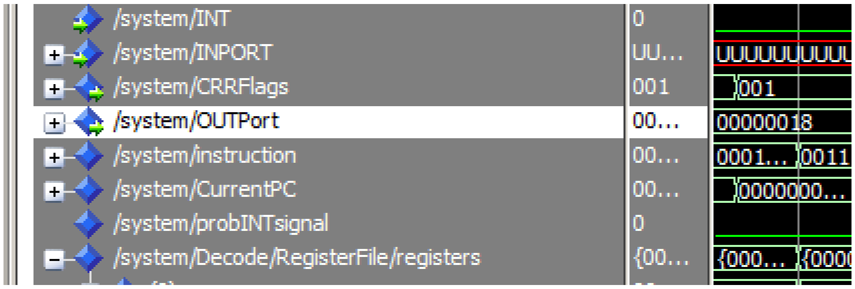
```
/system/Decode/RegisterFile/registers   {00...   {00000003} {
  (0)                                   00...    00000003
  (1)                                   00...    00000050
  (2)                                   00...    00000008
  (3)                                   00...    00000060
  (4)                                   00...    00000018
  (5)                                   00...    00000000
  (6)                                   00...    00000000
  (7)                                   00...    00000000
/system/probRstSignal                   0
```

ADD R4,R4,R4 (4th iter.)



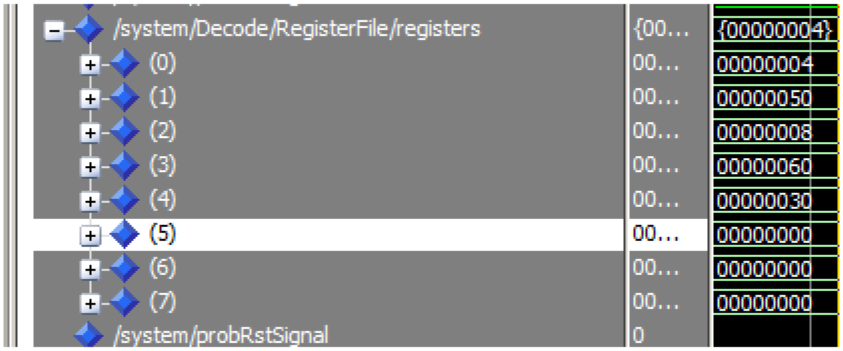
```
/system/Decode/RegisterFile/registers   {00...   {00000003} {0
  (0)                                   00...    00000003
  (1)                                   00...    00000050
  (2)                                   00...    00000008
  (3)                                   00...    00000060
  (4)                                   00...    00000030
  (5)                                   00...    00000000
  (6)                                   00...    00000000
  (7)                                   00...    00000000
/system/probRstSignal                   0
```

OUT R4



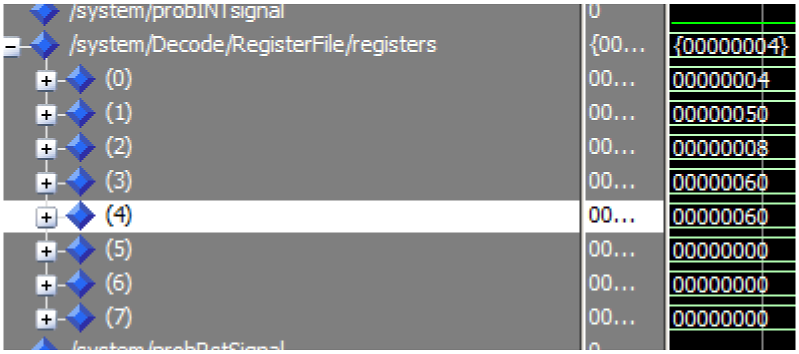
```
/system/INT                             0
/system/INPORT                          UU...    UUUUUUUUUU
/system/CRRFlags                        001      001
/system/OUTPort                         00...    00000018
/system/instruction                     00...    0001...  0011
/system/CurrentPC                       00...    0000000...
/system/probINTsignal                   0
/system/Decode/RegisterFile/registers   {00...   {000...  {0000
```

INC R0, AND R0,R2,R5 , JZ R3



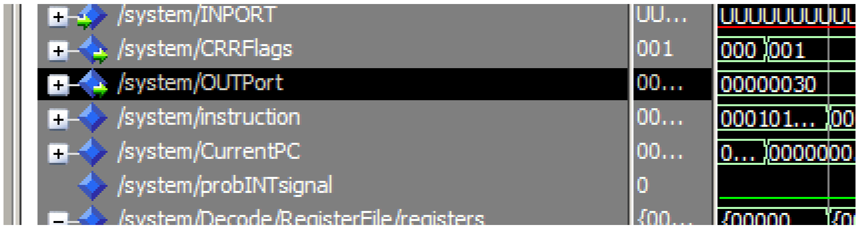
```
/system/Decode/RegisterFile/registers   {00...   {00000004}
  (0)                                   00...    00000004
  (1)                                   00...    00000050
  (2)                                   00...    00000008
  (3)                                   00...    00000060
  (4)                                   00...    00000030
  (5)                                   00...    00000000
  (6)                                   00...    00000000
  (7)                                   00...    00000000
/system/probRstSignal                   0
```

ADD R4,R4,R4   (4th iter.)



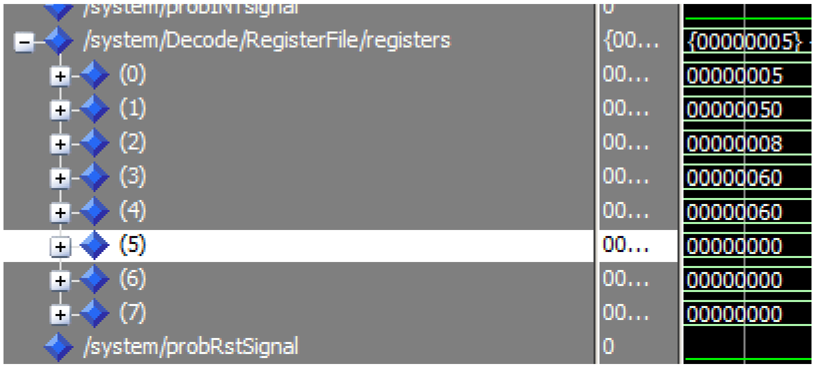
```
/system/probINTsignal                      0
/system/Decode/RegisterFile/registers      {00...   {00000004}
  (0)                                      00...    00000004
  (1)                                      00...    00000050
  (2)                                      00...    00000008
  (3)                                      00...    00000060
  (4)                                      00...    00000060
  (5)                                      00...    00000000
  (6)                                      00...    00000000
  (7)                                      00...    00000000
/system/probRstSignal                      0
```

OUT R4



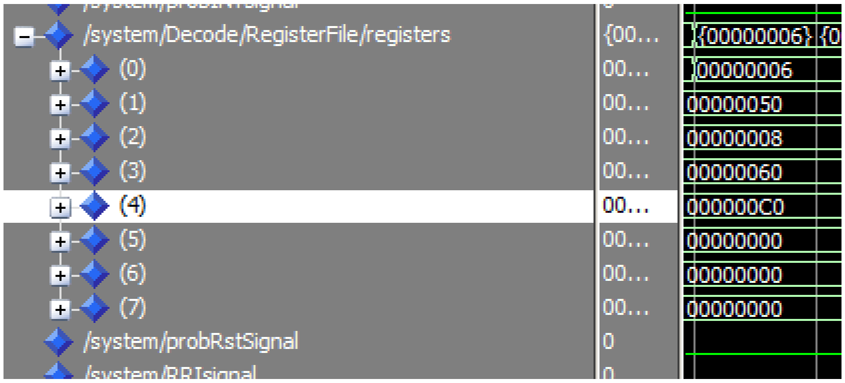
```
/system/INPORT                             UU...    UUUUUUUUUU
/system/CRRFlags                           001      000  001
/system/OUTPort                            00...    00000030
/system/instruction                        00...    000101...  00
/system/CurrentPC                          00...    0...  0000000
/system/probINTsignal                      0
/system/Decode/RegisterFile/registers      {00...   {00000    {0
```

INC R0, AND R0,R2,R5 , JZ R3

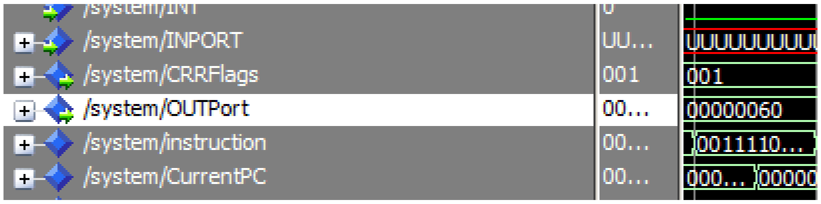


```
/system/probINTsignal                      0
/system/Decode/RegisterFile/registers      {00...   {00000005}
  (0)                                      00...    00000005
  (1)                                      00...    00000050
  (2)                                      00...    00000008
  (3)                                      00...    00000060
  (4)                                      00...    00000060
  (5)                                      00...    00000000
  (6)                                      00...    00000000
  (7)                                      00...    00000000
/system/probRstSignal                      0
```

ADD R4,R4,R4 ,OUT R4, INC R0,AND R0,R2,R5, JZ R3 (5th iter.)

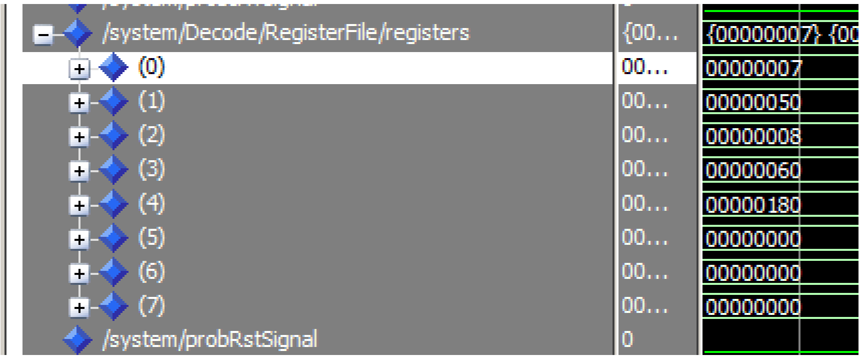


```
/system/probINTsignal                      0
/system/Decode/RegisterFile/registers      {00...   {00000006}  {0
  (0)                                      00...    00000006
  (1)                                      00...    00000050
  (2)                                      00...    00000008
  (3)                                      00...    00000060
  (4)                                      00...    000000C0
  (5)                                      00...    00000000
  (6)                                      00...    00000000
  (7)                                      00...    00000000
/system/probRstSignal                      0
/system/RRIsignal                          0
```

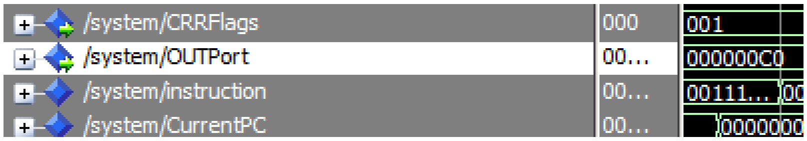


```
/system/INT                      0
/system/INPORT                   UU...   UUUUUUUUUU
/system/CRRFlags                 001     001
/system/OUTPort                  00...   00000060
/system/instruction              00...   0011110...
/system/CurrentPC                00...   000...  00000
```

ADD R4,R4,R4 ,OUT R4, INC R0,AND R0,R2,R5, JZ R3   (6th iter.)

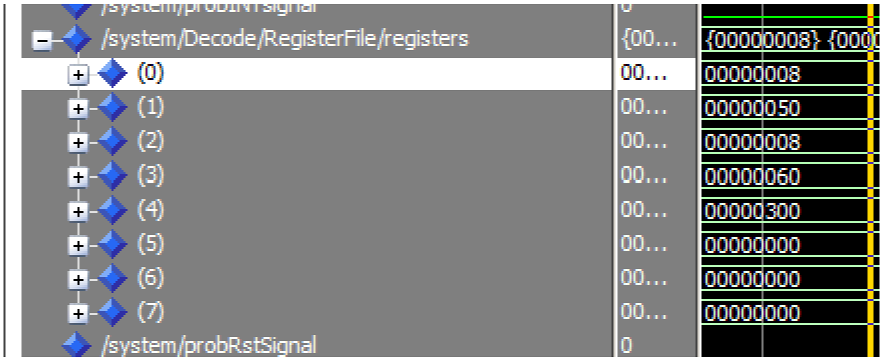


```
/system/Decode/RegisterFile/registers   {00...   {00000007} {00
  (0)                                   00...    00000007
  (1)                                   00...    00000050
  (2)                                   00...    00000008
  (3)                                   00...    00000060
  (4)                                   00...    00000180
  (5)                                   00...    00000000
  (6)                                   00...    00000000
  (7)                                   00...    00000000
/system/probRstSignal                   0
```

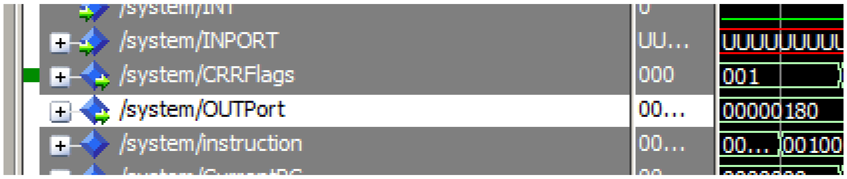


```
/system/CRRFlags                 000     001
/system/OUTPort                  00...   000000C0
/system/instruction              00...   00111...  00
/system/CurrentPC                00...   0000000
```

ADD R4,R4,R4 ,OUT R4, INC R0,AND R0,R2,R5, JZ R3 (7th iter.)

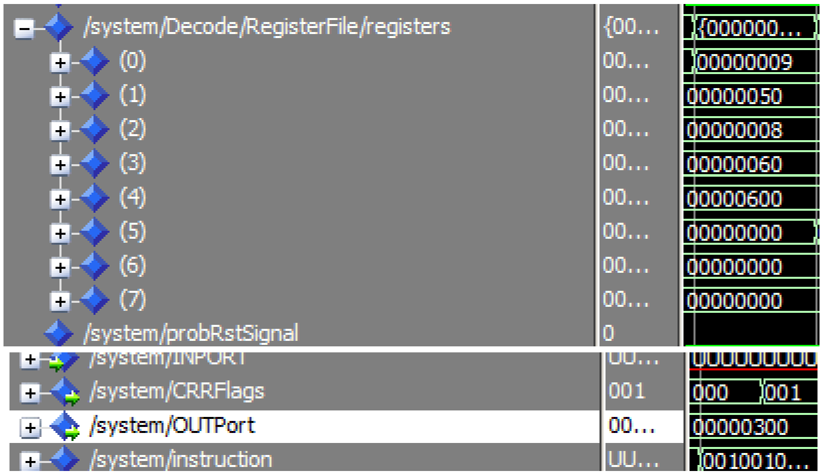


```
/system/probINTsignal                   0
/system/Decode/RegisterFile/registers   {00...   {00000008} {000
  (0)                                   00...    00000008
  (1)                                   00...    00000050
  (2)                                   00...    00000008
  (3)                                   00...    00000060
  (4)                                   00...    00000300
  (5)                                   00...    00000000
  (6)                                   00...    00000000
  (7)                                   00...    00000000
/system/probRstSignal                   0
```



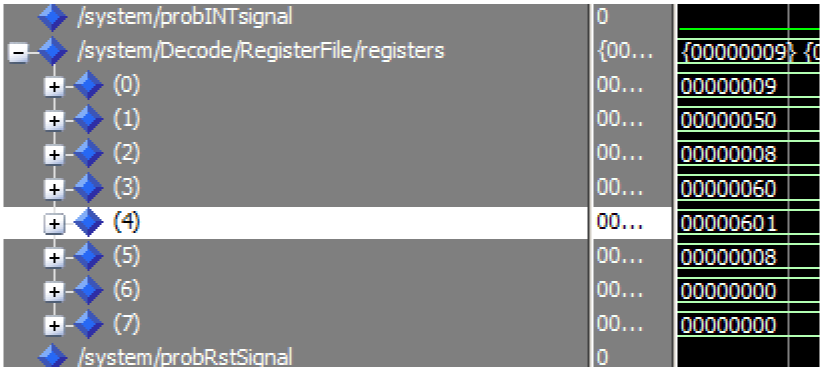
```
/system/INT                      0
/system/INPORT                   UU...   UUUUUUUUU
/system/CRRFlags                 000     001
/system/OUTPort                  00...   00000180
/system/instruction              00...   00...  00100
/system/CurrentPC                00...   0000000
```

ADD R4,R4,R4 ,OUT R4, INC R0,AND R0,R2,R5 ,JZ R3 (8th iter.)



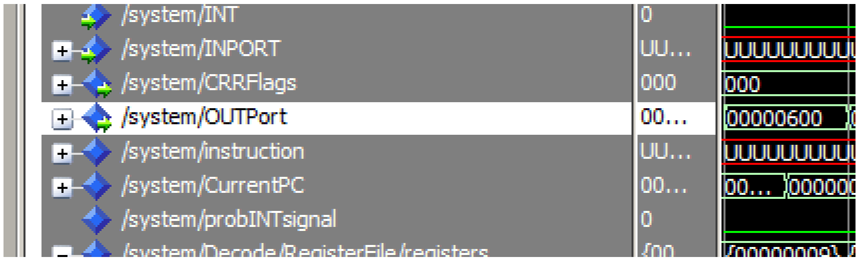
```
/system/Decode/RegisterFile/registers   {00...   {000000...
  (0)                                   00...    00000009
  (1)                                   00...    00000050
  (2)                                   00...    00000008
  (3)                                   00...    00000060
  (4)                                   00...    00000600
  (5)                                   00...    00000000
  (6)                                   00...    00000000
  (7)                                   00...    00000000
/system/probRstSignal                   0
/system/INPORT                          UU...    UUUUUUUUU
/system/CRRFlags                        001      000   001
/system/OUTPort                         00...    00000300
/system/instruction                     UU...    0010010...
```

INC R4



```
/system/probINTsignal                   0
/system/Decode/RegisterFile/registers   {00...   {00000009} {0
  (0)                                   00...    00000009
  (1)                                   00...    00000050
  (2)                                   00...    00000008
  (3)                                   00...    00000060
  (4)                                   00...    00000601
  (5)                                   00...    00000008
  (6)                                   00...    00000000
  (7)                                   00...    00000000
/system/probRstSignal                   0
```

OUT R4



```
/system/INT                             0
/system/INPORT                          UU...    UUUUUUUUU
/system/CRRFlags                        000      000
/system/OUTPort                         00...    00000600
/system/instruction                     UU...    UUUUUUUUU
/system/CurrentPC                       00...    00...  00000
/system/probINTsignal                   0
/system/Decode/RegisterFile/registers   {00...   {00000009} {
```

Solve by adding NOPs

Instruction	Hazard	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
LDM R2,0A		**F**	**D**	**E**	**M**	**W**																	
LDM R0,0			**F**	**D**	**E**	**M**	**W**																
LDM R1,50				**F**	**D**	**E**	**M**	**W**															
LDM R3,20					**F**	**D**	**E**	**M**	**W**														
LDM R4,2						**F**	**D**	**E**	**M**	**W**													
JMP R3							**F**	**D**	**E**	**M**	**W**												
SUB R0,R2,R5								**F**	**D**	**E**	**M**	**W**											
JZ R1									**F**	**D**	**E**	**M**	**W**										
ADD R4,R4,R4										**F**	**D**	**E**	**M**	**W**									

Instruction	Hazards	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
NOP		**F**	**D**	**E**	**M**	**W**													
NOP			**F**	**D**	**E**	**M**	**W**												

OUT R4	RAW R4 Data Hazard			F	D	E	M	W											
INC R0					F	D	E	M	W										
JMP R3 (1St iter.)						F	D	E	M	W									
NOP							F	D	E	M	W								
SUB R0,R2,R5	RAW R0 Data Hazard							F	D	E	M	W							
JZ R1									F	D	E	M	W						
ADD R4,R4,R4										F	D	E	M	W					
NOP											F	D	E	M	W				
NOP												F	D	E	M	W			
OUT R4	RAW R4 Data Hazard												F	D	E	M	W		
INC R0														F	D	E	M	W	
JMP R3 (2nd iter.)															F	D	E	M	W

Instruction	Hazard	24	25	26	27	28	29	30	31	32	33	34	35	36
NOP		F	D	E	M	W								
SUB R0,R2,R5	RAW R0 Data Hazard		F	D	E	M	W							
JZ R1				F	D	E	M	W						
ADD R4,R4,R4					F	D	E	M	W					
NOP						F	D	E	M	W				
NOP							F	D	E	M	W			
OUT R4	RAW R4 Data Hazard							F	D	E	M	W		
INC R0									F	D	E	M	W	
JMP R3 (3rd iter.)										F	D	E	M	W

Instruction	Hazard	34	35	36	37	38	39	40	41	42	43	44	45	46
NOP		F	D	E	M	W								
SUB R0,R2,R5	RAW R0 Data Hazard		F	D	E	M	W							

JZ R1				F	D	E	M	W						
ADD R4,R4,R4					F	D	E	M	W					
NOP						F	D	E	M	W				
NOP							F	D	E	M	W			
OUT R4	RAW R4 Data Hazard							F	D	E	M	W		
INC R0									F	D	E	M	W	
JMP R3 (4th iter.)										F	D	E	M	W

Instruction	Hazard	43	44	45	46	47	48	49	50	51	52	53	54	55
NOP		F	D	E	M	W								
SUB R0,R2,R5	RAW R0 Data Hazard		F	D	E	M	W							
JZ R1				F	D	E	M	W						
ADD R4,R4,R4					F	D	E	M	W					
NOP						F	D	E	M	W				

NOP							F	D	E	M	W			
OUT R4	RAW R4 Data Hazard							F	D	E	M	W		
INC R0									F	D	E	M	W	
JMP R3 (5th iter.)										F	D	E	M	W

Instruction	Hazard	52	53	54	55	56	57	58	59	60	61	62	63	64
NOP		F	D	E	M	W								
SUB R0,R2,R5	RAW R0 Data Hazard		F	D	E	M	W							
JZ R1				F	D	E	M	W						
ADD R4,R4,R4					F	D	E	M	W					
NOP						F	D	E	M	W				
NOP							F	D	E	M	W			
OUT R4	RAW R4 Data Hazard							F	D	E	M	W		
INC R0									F	D	E	M	W	

JMP R3 (6th iter.)										F	D	E	M	W

Instruction	Hazard	61	62	63	64	65	66	67	68	69	70	71	72	73
NOP		F	D	E	M	W								
SUB R0,R2,R5	RAW R0 Data Hazard		F	D	E	M	W							
JZ R1				F	D	E	M	W						
ADD R4,R4,R4					F	D	E	M	W					
NOP						F	D	E	M	W				
NOP							F	D	E	M	W			
OUT R4	RAW R4 Data Hazard							F	D	E	M	W		
INC R0									F	D	E	M	W	
JMP R3 (7th iter.)										F	D	E	M	W

Instruction	Hazard	70	71	72	73	74	75	76	77	78	79	80	81	82
NOP		F	D	E	M	W								
SUB R0,R2,R5	RAW R0 Data Hazard		F	D	E	M	W							

JZ R1				F	D	E	M	W						
ADD R4,R4,R4					F	D	E	M	W					
NOP						F	D	E	M	W				
NOP							F	D	E	M	W			
OUT R4	RAW R4 Data Hazard							F	D	E	M	W		
INC R0									F	D	E	M	W	
JMP R3 (8th iter.)										F	D	E	M	W

Instruction	Hazard	79	80	81	82	83	84	85	86	87	88	89	90	91
NOP		F	D	E	M	W								
SUB R0,R2,R5	RAW R0 Data Hazard		F	D	E	M	W							
JZ R1				F	D	E	M	W						
ADD R4,R4,R4					F	D	E	M	W					
NOP						F	D	E	M	W				
NOP							F	D	E	M	W			

OUT R4	RAW R4 Data Hazard							F	D	E	M	W		
INC R0									F	D	E	M	W	
JMP R3 (9th iter.)										F	D	E	M	W

Instruction	Hazard	88	89	90	91	92	93	94	95	96	97	98	99	100
NOP		F	D	E	M	W								
SUB R0,R2,R5	RAW R0 Data Hazard		F	D	E	M	W							
JZ R1				F	D	E	M	W						
ADD R4,R4,R4					F	D	E	M	W					
NOP						F	D	E	M	W				
NOP							F	D	E	M	W			
OUT R4	RAW R4 Data Hazard							F	D	E	M	W		
INC R0									F	D	E	M	W	
JMP R3 (10th iter.)										F	D	E	M	W

Instruction	Hazard	97	98	99	100	101	102	103	104	105	106	107	108	109
LDM R0,0		F	D	E	M	W								
LDM R2,8			F	D	E	M	W							
LDM R3,60				F	D	E	M	W						
LDM R4,3					F	D	E	M	W					
JMP R3						F	D	E	M	W				
ADD R4, R4,R4							F	D	E	M	W			
NOP								F	D	E	M	W		
NOP									F	D	E	M	W	
OUT R4	RAW R4 Data Hazard									F	D	E	M	W

Instruction	Hazard	106	107	108	109	110	112	113	114	115	116	117	118	119
INC R0		F	D	E	M	W								
NOP			F	D	E	M	W							

NOP				F	D	E	M	W						
AND R0,R2,R5	RAW R0 Data Hazard				F	D	E	M	W					
JZ R3 1st iter.						F	D	E	M	W				
ADD R4, R4,R4							F	D	E	M	W			
NOP								F	D	E	M	W		
NOP									F	D	E	M	W	
OUT R4	RAW R4 Data Hazard									F	D	E	M	W

Instruction	Hazard	116	117	118	119	120	122	123	124	125	126
INC R0			F	D	E	M	W				
NOP				F	D	E	M	W			
NOP					F	D	E	M	W		
AND R0,R2,R5	RAW R0 Data Hazard					F	D	E	M	W	
JZ R3 (2nd iter.)							F	D	E	M	W

Instruction	Hazard	123	124	125	126	127	128	129	130	131	132	133	134	135
ADD R4, R4,R4		F	D	E	M	W								
NOP			F	D	E	M	W							
NOP				F	D	E	M	W						
OUT R4	RAW R4 Data Hazard				F	D	E	M	W					
INC R0						F	D	E	M	W				
NOP							F	D	E	M	W			
NOP								F	D	E	M	W		
AND R0,R2,R5	RAW R0 Data Hazard								F	D	E	M	W	
JZ R3 (3rd iter.)										F	D	E	M	W

Instruction	Hazard	132	133	134	135	136	137	138	139	140	141	142	143	144
ADD R4, R4,R4		F	D	E	M	W								
NOP			F	D	E	M	W							

NOP				F	D	E	M	W						
OUT R4	RAW R4 Data Hazard				F	D	E	M	W					
INC R0						F	D	E	M	W				
NOP							F	D	E	M	W			
NOP								F	D	E	M	W		
AND R0,R2,R5	RAW R0 Data Hazard								F	D	E	M	W	
JZ R3 (4th iter.)										F	D	E	M	W

Instruction	Hazard	141	142	143	144	145	146	147	148	149	150	151	152	153
ADD R4, R4,R4		F	D	E	M	W								
NOP			F	D	E	M	W							
NOP				F	D	E	M	W						
OUT R4	RAW R4 Data Hazard				F	D	E	M	W					
INC R0						F	D	E	M	W				

NOP							F	D	E	M	W			
NOP								F	D	E	M	W		
AND R0,R2,R5	RAW R0 Data Hazard								F	D	E	M	W	
JZ R3 (5th iter.)										F	D	E	M	W

Instruction	Hazard	150	151	152	153	154	155	156	157	158	159	160	161	162
ADD R4, R4,R4		F	D	E	M	W								
NOP			F	D	E	M	W							
NOP				F	D	E	M	W						
OUT R4	RAW R4 Data Hazard				F	D	E	M	W					
INC R0						F	D	E	M	W				
NOP							F	D	E	M	W			
NOP								F	D	E	M	W		
AND R0,R2,R5	RAW R0 Data Hazard								F	D	E	M	W	

JZ R3 (6th iter.)										F	D	E	M	W

Instruction	Hazard	159	160	161	162	163	164	165	166	167	168	169	170	171
ADD R4, R4,R4		F	D	E	M	W								
NOP			F	D	E	M	W							
NOP				F	D	E	M	W						
OUT R4	RAW R4 Data Hazard				F	D	E	M	W					
INC R0						F	D	E	M	W				
NOP							F	D	E	M	W			
NOP								F	D	E	M	W		
AND R0,R2,R5	RAW R0 Data Hazard								F	D	E	M	W	
JZ R3 (7th iter.)										F	D	E	M	W

Instruction	Hazard	169	170	171	172	173	174	175	176	177	178	179	180	181
ADD R4, R4,R4		F	D	E	M	W								
NOP			F	D	E	M	W							
NOP				F	D	E	M	W						
OUT R4	RAW R4 Data Hazard				F	D	E	M	W					
INC R0						F	D	E	M	W				
NOP							F	D	E	M	W			
NOP								F	D	E	M	W		
AND R0,R2,R5	RAW R0 Data Hazard								F	D	E	M	W	
JZ R3 (8th iter.)										F	D	E	M	W

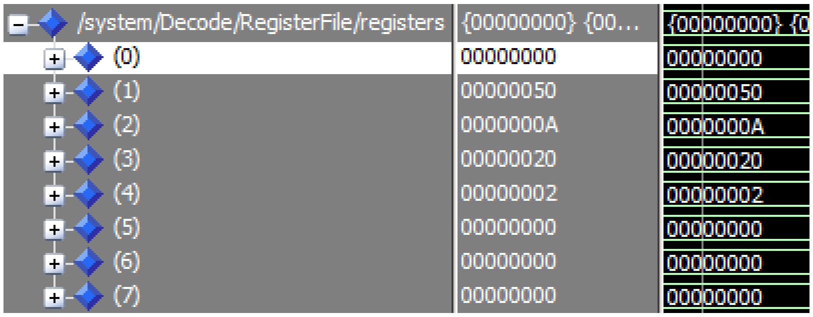
Instruction	Hazard	178	179	180	181	182	183	184	185
INC R4		F	D	E	M	W			
NOP			F	D	E	M	W		

NOP				F	D	E	M	W	
OUT R4	**RAW R4** **Data Hazard**				**F**	**D**	**E**	**M**	**W**

Here is the correct result with forwarding unit only
All instructions works CORRECTLY and give the right results with no hazards because all hazards are data hazards that are handled by forwarding unit.

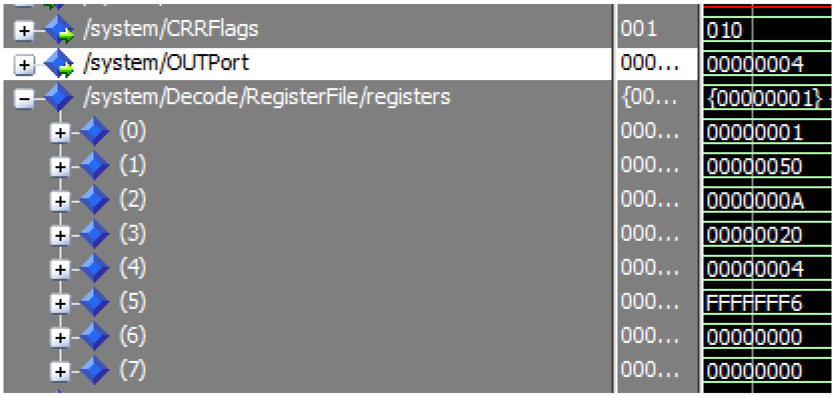
The following wave shows the register file

LDM R2,0A, LDM R0,0 , LDM R1,50, LDM R3,20, LDM R4,2 , JMP R3



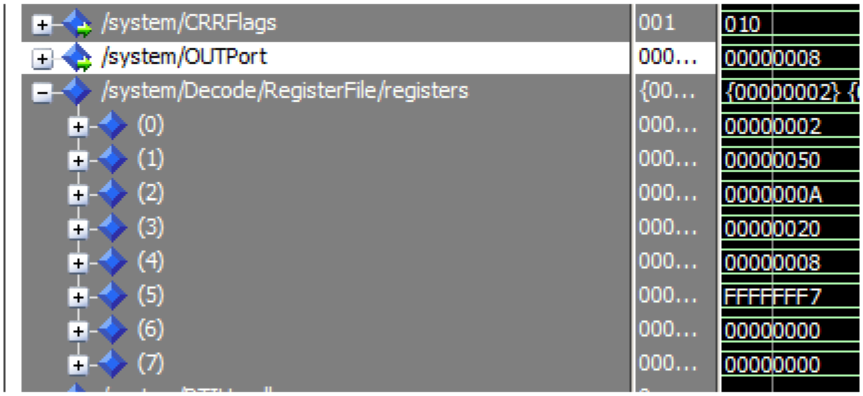
```
/system/Decode/RegisterFile/registers   {00000000} {00...   {00000000} {0
  (0)                                   00000000            00000000
  (1)                                   00000050            00000050
  (2)                                   0000000A            0000000A
  (3)                                   00000020            00000020
  (4)                                   00000002            00000002
  (5)                                   00000000            00000000
  (6)                                   00000000            00000000
  (7)                                   00000000            00000000
```

SUB R0,R2,R5 ,JZ R1, ADD R4,R4,R4 ,OUT R4 ,INC R0, JMP R3 (1st iteration)



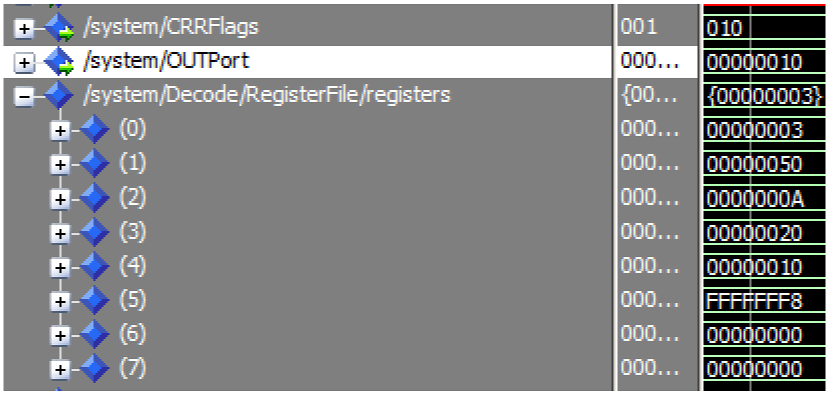
```
/system/CRRFlags                        001      010
/system/OUTPort                         000...   00000004
/system/Decode/RegisterFile/registers   {00...   {00000001}
  (0)                                   000...   00000001
  (1)                                   000...   00000050
  (2)                                   000...   0000000A
  (3)                                   000...   00000020
  (4)                                   000...   00000004
  (5)                                   000...   FFFFFFF6
  (6)                                   000...   00000000
  (7)                                   000...   00000000
```

SUB R0,R2,R5 ,JZ R1, ADD R4,R4,R4 ,OUT R4 ,INC R0, JMP R3 (2nd iteration)



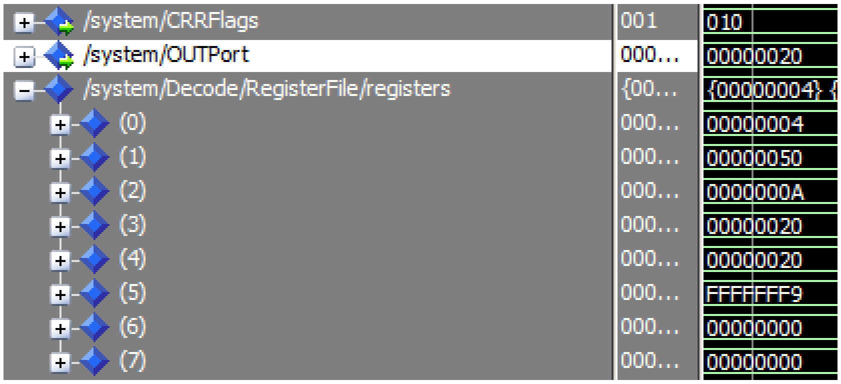
```
/system/CRRFlags                         001     010
/system/OUTPort                          000...  00000008
/system/Decode/RegisterFile/registers    {00...  {00000002} {
    (0)                                  000...  00000002
    (1)                                  000...  00000050
    (2)                                  000...  0000000A
    (3)                                  000...  00000020
    (4)                                  000...  00000008
    (5)                                  000...  FFFFFFF7
    (6)                                  000...  00000000
    (7)                                  000...  00000000
```

SUB R0,R2,R5 ,JZ R1, ADD R4,R4,R4 ,OUT R4 ,INC R0, JMP R3 (3rd iteration)



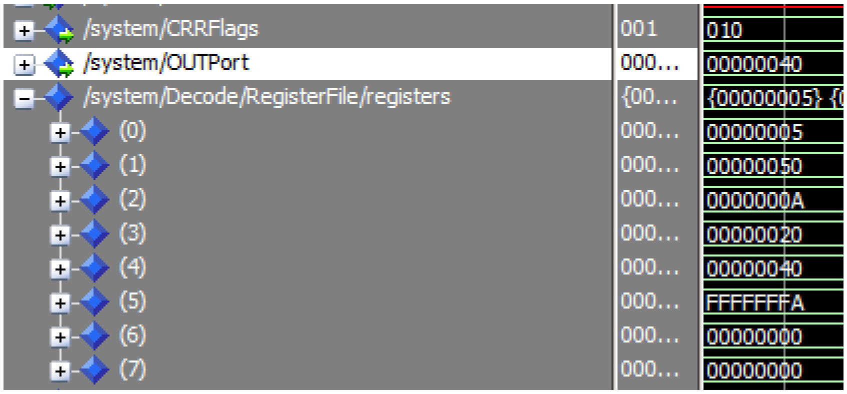
```
/system/CRRFlags                         001     010
/system/OUTPort                          000...  00000010
/system/Decode/RegisterFile/registers    {00...  {00000003}
    (0)                                  000...  00000003
    (1)                                  000...  00000050
    (2)                                  000...  0000000A
    (3)                                  000...  00000020
    (4)                                  000...  00000010
    (5)                                  000...  FFFFFFF8
    (6)                                  000...  00000000
    (7)                                  000...  00000000
```

SUB R0,R2,R5 ,JZ R1, ADD R4,R4,R4 ,OUT R4 ,INC R0, JMP R3 (4th iteration)



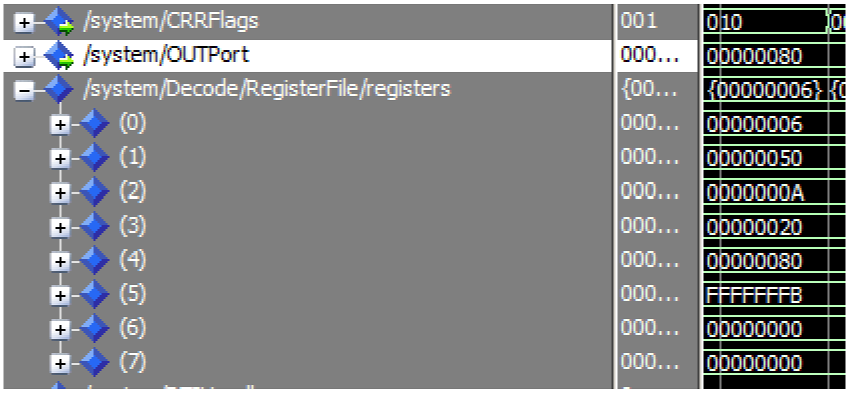
```
/system/CRRFlags                         001     010
/system/OUTPort                          000...  00000020
/system/Decode/RegisterFile/registers    {00...  {00000004} {
    (0)                                  000...  00000004
    (1)                                  000...  00000050
    (2)                                  000...  0000000A
    (3)                                  000...  00000020
    (4)                                  000...  00000020
    (5)                                  000...  FFFFFFF9
    (6)                                  000...  00000000
    (7)                                  000...  00000000
```

SUB R0,R2,R5 ,JZ R1, ADD R4,R4,R4 ,OUT R4 ,INC R0, JMP R3 (5th iteration)



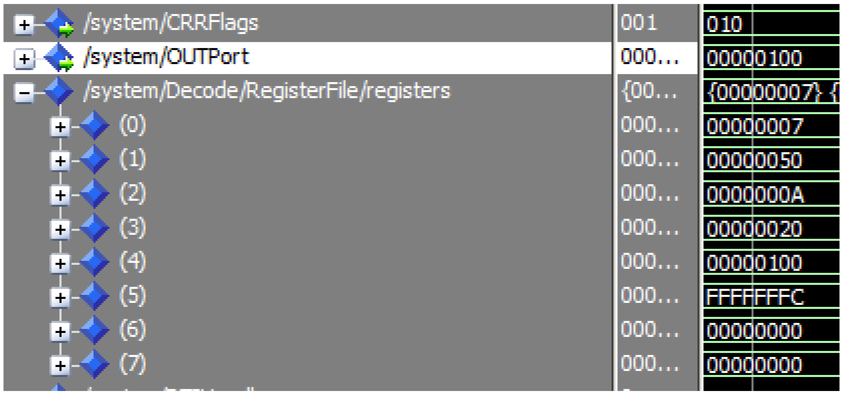
```
/system/CRRFlags                      001    010
/system/OUTPort                       000... 00000040
/system/Decode/RegisterFile/registers {00... {00000005} {0
  (0)                                 000... 00000005
  (1)                                 000... 00000050
  (2)                                 000... 0000000A
  (3)                                 000... 00000020
  (4)                                 000... 00000040
  (5)                                 000... FFFFFFFA
  (6)                                 000... 00000000
  (7)                                 000... 00000000
```

SUB R0,R2,R5 ,JZ R1, ADD R4,R4,R4 ,OUT R4 ,INC R0, JMP R3 (6th iteration)



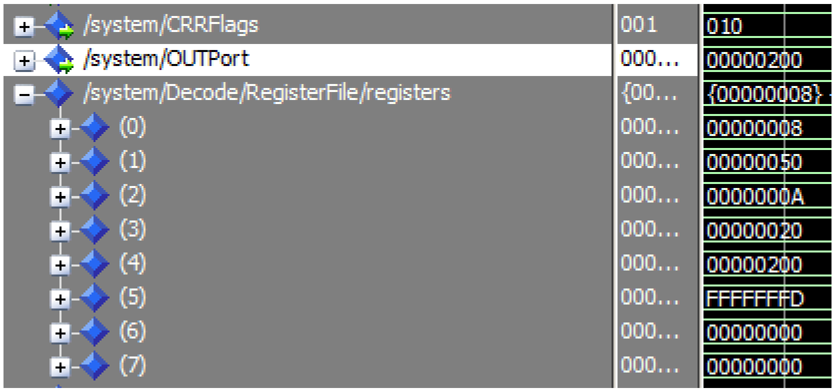
```
/system/CRRFlags                      001    010
/system/OUTPort                       000... 00000080
/system/Decode/RegisterFile/registers {00... {00000006} {0
  (0)                                 000... 00000006
  (1)                                 000... 00000050
  (2)                                 000... 0000000A
  (3)                                 000... 00000020
  (4)                                 000... 00000080
  (5)                                 000... FFFFFFFB
  (6)                                 000... 00000000
  (7)                                 000... 00000000
```

SUB R0,R2,R5 ,JZ R1, ADD R4,R4,R4 ,OUT R4 ,INC R0, JMP R3 (7th iteration)



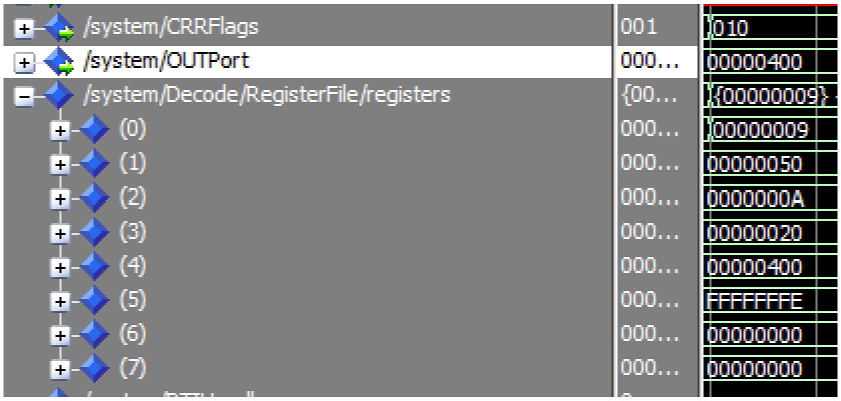
```
/system/CRRFlags                      001    010
/system/OUTPort                       000... 00000100
/system/Decode/RegisterFile/registers {00... {00000007} {
  (0)                                 000... 00000007
  (1)                                 000... 00000050
  (2)                                 000... 0000000A
  (3)                                 000... 00000020
  (4)                                 000... 00000100
  (5)                                 000... FFFFFFFC
  (6)                                 000... 00000000
  (7)                                 000... 00000000
```

SUB R0,R2,R5 ,JZ R1, ADD R4,R4,R4 ,OUT R4 ,INC R0, JMP R3 (8th iteration)



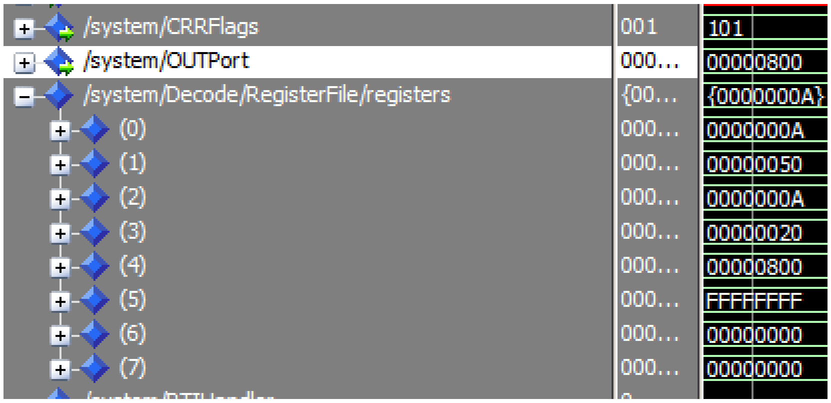
```
/system/CRRFlags                          001     010
/system/OUTPort                           000...  00000200
/system/Decode/RegisterFile/registers     {00...  {00000008}
  (0)                                     000...  00000008
  (1)                                     000...  00000050
  (2)                                     000...  0000000A
  (3)                                     000...  00000020
  (4)                                     000...  00000200
  (5)                                     000...  FFFFFFFD
  (6)                                     000...  00000000
  (7)                                     000...  00000000
```

SUB R0,R2,R5 ,JZ R1, ADD R4,R4,R4 ,OUT R4 ,INC R0, JMP R3 (9th iteration)



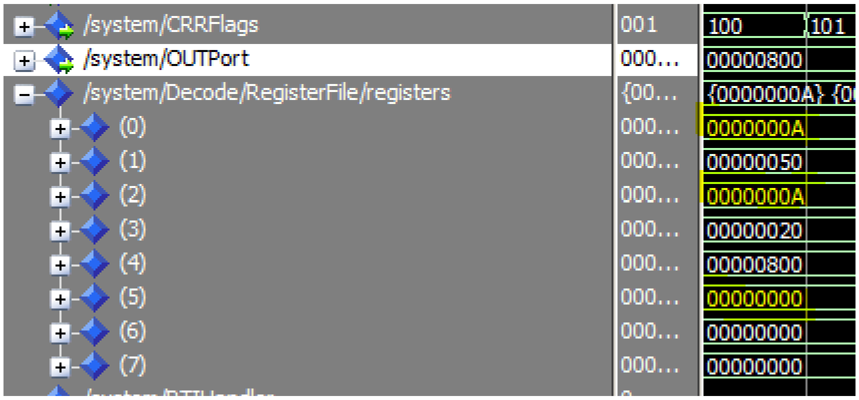
```
/system/CRRFlags                          001     010
/system/OUTPort                           000...  00000400
/system/Decode/RegisterFile/registers     {00...  {00000009}
  (0)                                     000...  00000009
  (1)                                     000...  00000050
  (2)                                     000...  0000000A
  (3)                                     000...  00000020
  (4)                                     000...  00000400
  (5)                                     000...  FFFFFFFE
  (6)                                     000...  00000000
  (7)                                     000...  00000000
```

SUB R0,R2,R5 ,JZ R1, ADD R4,R4,R4 ,OUT R4 ,INC R0, JMP R3 (10th iteration)



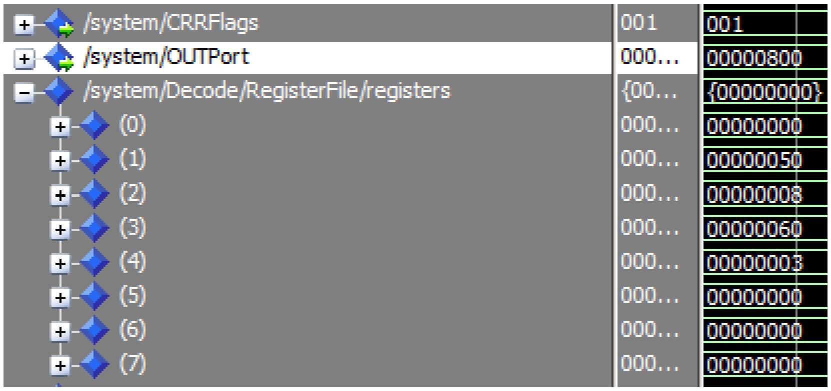
```
/system/CRRFlags                          001     101
/system/OUTPort                           000...  00000800
/system/Decode/RegisterFile/registers     {00...  {0000000A}
  (0)                                     000...  0000000A
  (1)                                     000...  00000050
  (2)                                     000...  0000000A
  (3)                                     000...  00000020
  (4)                                     000...  00000800
  (5)                                     000...  FFFFFFFF
  (6)                                     000...  00000000
  (7)                                     000...  00000000
```

SUB R0,R2,R5 ,JZ R1



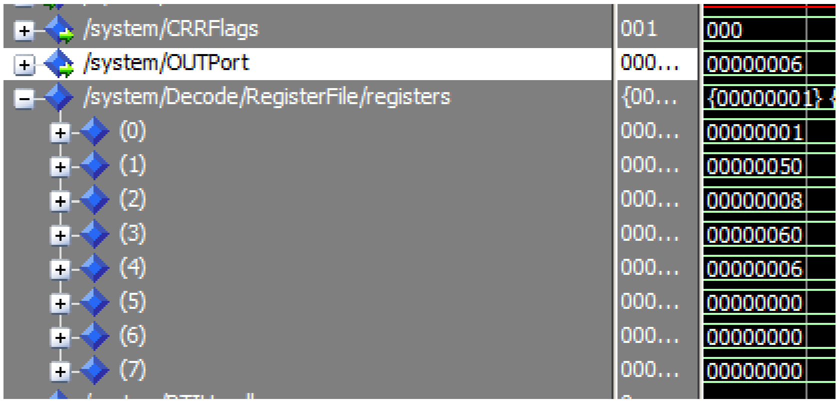
```
+ /system/CRRFlags                          001    100       101
+ /system/OUTPort                           000...  00000800
- /system/Decode/RegisterFile/registers     {00...  {0000000A} {0(
   + (0)                                    000...  0000000A
   + (1)                                    000...  00000050
   + (2)                                    000...  0000000A
   + (3)                                    000...  00000020
   + (4)                                    000...  00000800
   + (5)                                    000...  00000000
   + (6)                                    000...  00000000
   + (7)                                    000...  00000000
```

LDM R0,0 , LDM R2,8 , LDM R3,60 , LDM R4,3 , JMP R3



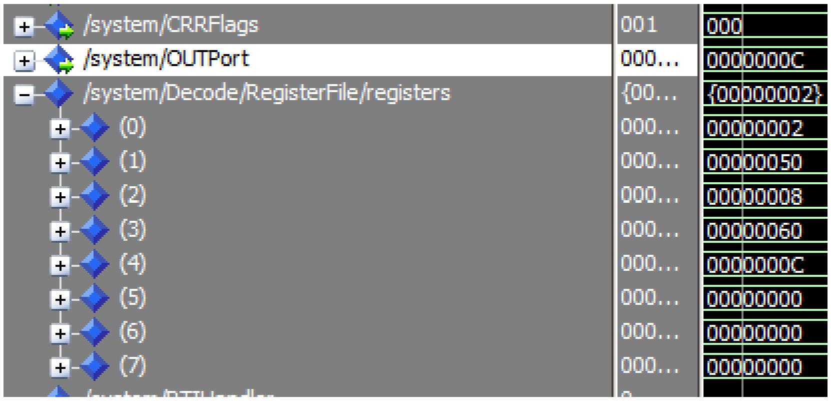
```
+ /system/CRRFlags                          001    001
+ /system/OUTPort                           000...  00000800
- /system/Decode/RegisterFile/registers     {00...  {00000000}
   + (0)                                    000...  00000000
   + (1)                                    000...  00000050
   + (2)                                    000...  00000008
   + (3)                                    000...  00000060
   + (4)                                    000...  00000003
   + (5)                                    000...  00000000
   + (6)                                    000...  00000000
   + (7)                                    000...  00000000
```

ADD R4,R4,R4 , OUT R4 ,INC R0, AND R0,R2,R5 , JZ R3 (1s iter.)



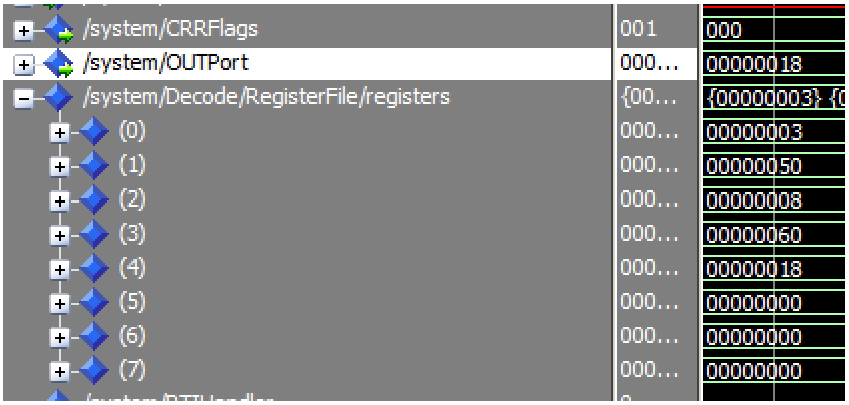
```
+ /system/CRRFlags                          001    000
+ /system/OUTPort                           000...  00000006
- /system/Decode/RegisterFile/registers     {00...  {00000001}
   + (0)                                    000...  00000001
   + (1)                                    000...  00000050
   + (2)                                    000...  00000008
   + (3)                                    000...  00000060
   + (4)                                    000...  00000006
   + (5)                                    000...  00000000
   + (6)                                    000...  00000000
   + (7)                                    000...  00000000
```

ADD R4,R4,R4 , OUT R4 ,INC R0, AND R0,R2,R5 , JZ R3 (2nd iter.)



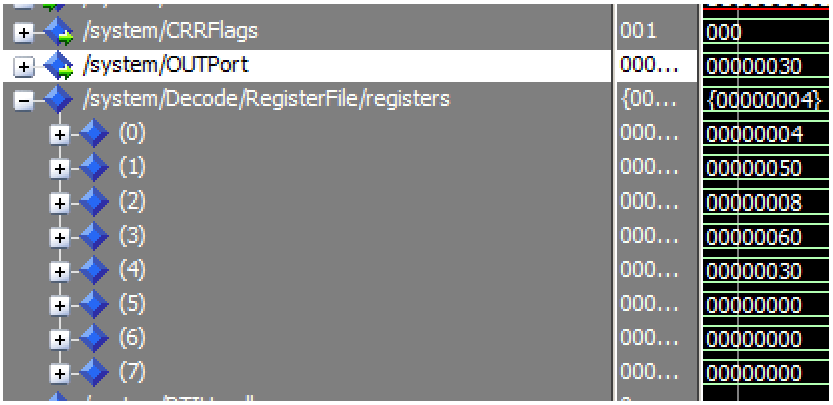
```
/system/CRRFlags                          001    000
/system/OUTPort                           000...  0000000C
/system/Decode/RegisterFile/registers     {00...  {00000002}
    (0)                                   000...  00000002
    (1)                                   000...  00000050
    (2)                                   000...  00000008
    (3)                                   000...  00000060
    (4)                                   000...  0000000C
    (5)                                   000...  00000000
    (6)                                   000...  00000000
    (7)                                   000...  00000000
/system/RTIHandler                        0
```

ADD R4,R4,R4 , OUT R4 ,INC R0, AND R0,R2,R5 , JZ R3 (3rd iter.)



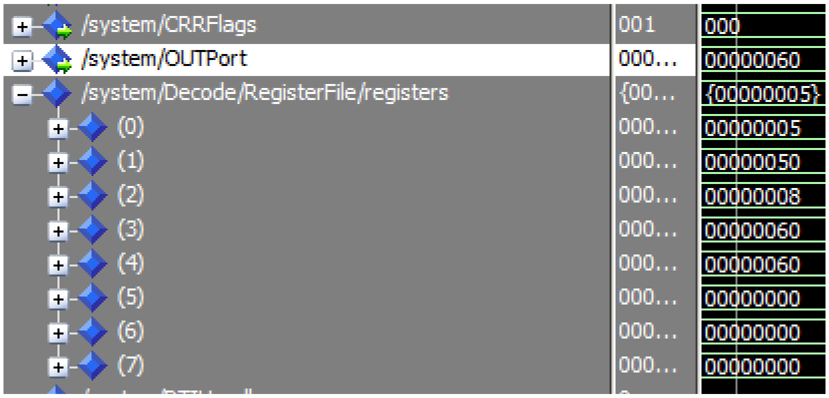
```
/system/CRRFlags                          001    000
/system/OUTPort                           000...  00000018
/system/Decode/RegisterFile/registers     {00...  {00000003} {0
    (0)                                   000...  00000003
    (1)                                   000...  00000050
    (2)                                   000...  00000008
    (3)                                   000...  00000060
    (4)                                   000...  00000018
    (5)                                   000...  00000000
    (6)                                   000...  00000000
    (7)                                   000...  00000000
/system/RTIHandler                        0
```

ADD R4,R4,R4 , OUT R4 ,INC R0, AND R0,R2,R5 , JZ R3 (4th iter.)



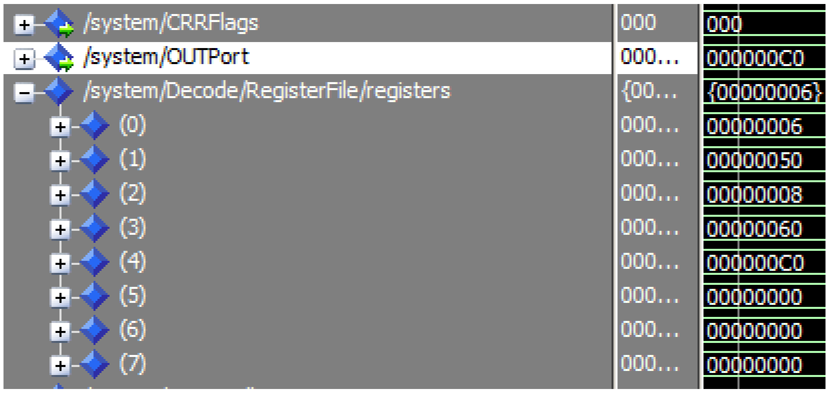
```
/system/CRRFlags                          001    000
/system/OUTPort                           000...  00000030
/system/Decode/RegisterFile/registers     {00...  {00000004}
    (0)                                   000...  00000004
    (1)                                   000...  00000050
    (2)                                   000...  00000008
    (3)                                   000...  00000060
    (4)                                   000...  00000030
    (5)                                   000...  00000000
    (6)                                   000...  00000000
    (7)                                   000...  00000000
/system/RTIHandler                        0
```

ADD R4,R4,R4 , OUT R4 ,INC R0, AND R0,R2,R5 , JZ R3 (5th iter.)



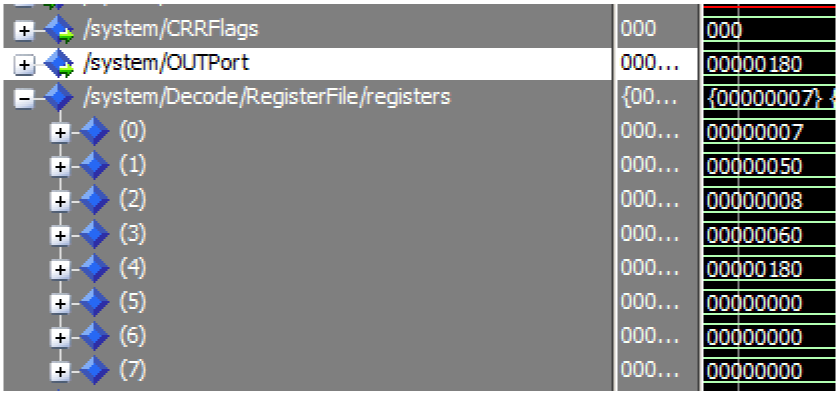
```
/system/CRRFlags                        001     000
/system/OUTPort                         000...  00000060
/system/Decode/RegisterFile/registers   {00...  {00000005}
    (0)                                 000...  00000005
    (1)                                 000...  00000050
    (2)                                 000...  00000008
    (3)                                 000...  00000060
    (4)                                 000...  00000060
    (5)                                 000...  00000000
    (6)                                 000...  00000000
    (7)                                 000...  00000000
```

ADD R4,R4,R4 , OUT R4 ,INC R0, AND R0,R2,R5 , JZ R3 (6th iter.)



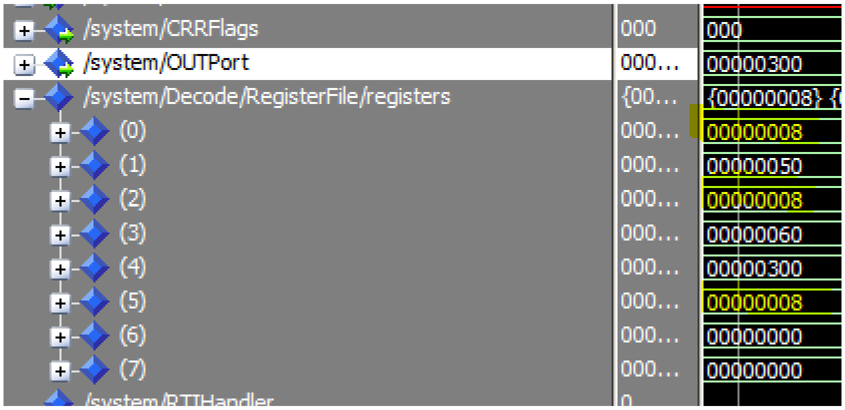
```
/system/CRRFlags                        000     000
/system/OUTPort                         000...  000000C0
/system/Decode/RegisterFile/registers   {00...  {00000006}
    (0)                                 000...  00000006
    (1)                                 000...  00000050
    (2)                                 000...  00000008
    (3)                                 000...  00000060
    (4)                                 000...  000000C0
    (5)                                 000...  00000000
    (6)                                 000...  00000000
    (7)                                 000...  00000000
```

ADD R4,R4,R4 , OUT R4 ,INC R0, AND R0,R2,R5 , JZ R3 (7th iter.)



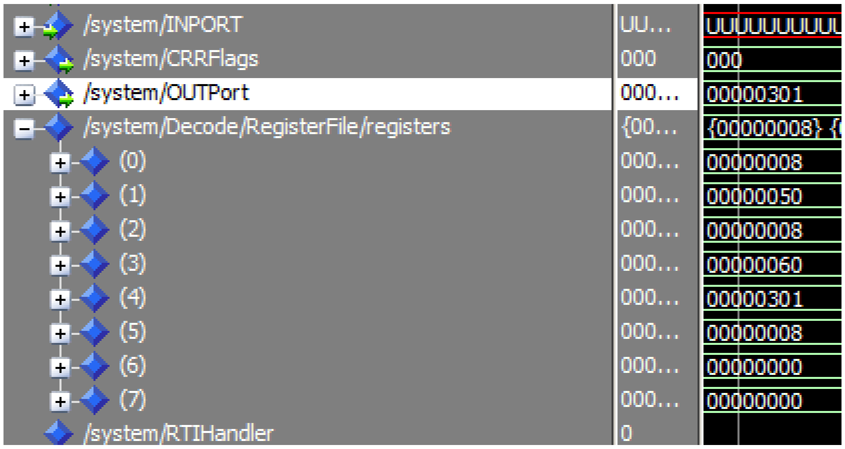
```
/system/CRRFlags                        000     000
/system/OUTPort                         000...  00000180
/system/Decode/RegisterFile/registers   {00...  {00000007} {
    (0)                                 000...  00000007
    (1)                                 000...  00000050
    (2)                                 000...  00000008
    (3)                                 000...  00000060
    (4)                                 000...  00000180
    (5)                                 000...  00000000
    (6)                                 000...  00000000
    (7)                                 000...  00000000
```

ADD R4,R4,R4 , OUT R4 ,INC R0, AND R0,R2,R5 , JZ R3 (8th iter.)



```
/system/CRRFlags                          000     000
/system/OUTPort                           000...  00000300
/system/Decode/RegisterFile/registers     {00...  {00000008} {(
    (0)                                   000...  00000008
    (1)                                   000...  00000050
    (2)                                   000...  00000008
    (3)                                   000...  00000060
    (4)                                   000...  00000300
    (5)                                   000...  00000008
    (6)                                   000...  00000000
    (7)                                   000...  00000000
/system/RTIHandler                        0
```

INC R4 OUT R4



```
/system/INPORT                            UU...   UUUUUUUUUU
/system/CRRFlags                          000     000
/system/OUTPort                           000...  00000301
/system/Decode/RegisterFile/registers     {00...  {00000008} {(
    (0)                                   000...  00000008
    (1)                                   000...  00000050
    (2)                                   000...  00000008
    (3)                                   000...  00000060
    (4)                                   000...  00000301
    (5)                                   000...  00000008
    (6)                                   000...  00000000
    (7)                                   000...  00000000
/system/RTIHandler                        0
```

Design

Instruction format

- Width of instruction = 16 bits
- 5 bits for opcode for all kind of instruction

Opcodes:

One operand

Opcode	Rdst	000	00000
5 bits	3 bits	3 bits	5 bits

Two operand (ADD, SUB, OR, AND)

Opcode	Rsrc1	Rscr2	Rdst	0
5 bits	3 bits	3bits	3 bits	2 bit

Two operand (SWAP)

Opcode	Rsrc1	Rscr2	000	0
5 bits	3 bits	3bits	3 bits	2 bit

Two operand (SHL, SHR)

Opcode	Rsrc	000	0000	1
5 bits	3 bits	3bits	4 bits	1 bit 32/16

IMM
16 bit

Two operand (IADD)

Opcode	Rsrc	000	RDst	01
5 bits	3 bits	3bits	3 bits	zero+1 bit 32/16

IMM
16 bit

Memory (PUSH, POP)

Opcode	Rdst	000	00000
5 bits	3 bits	3 bits	5 bit

Memory (LDM)

Opcode 5 bits	Rdst 3 bits	000 3 bits	0000 4 bit	1 1 bit 32/16

IMM 16 bit

Memory (LDD, STD)

Opcode 5 bits	Rdst 3 bits	Rsrc 3 bits	EA(19:16) 4 bit	1 1 bit

EA(0:15) 16 bit

Branch and Change of Control Operations

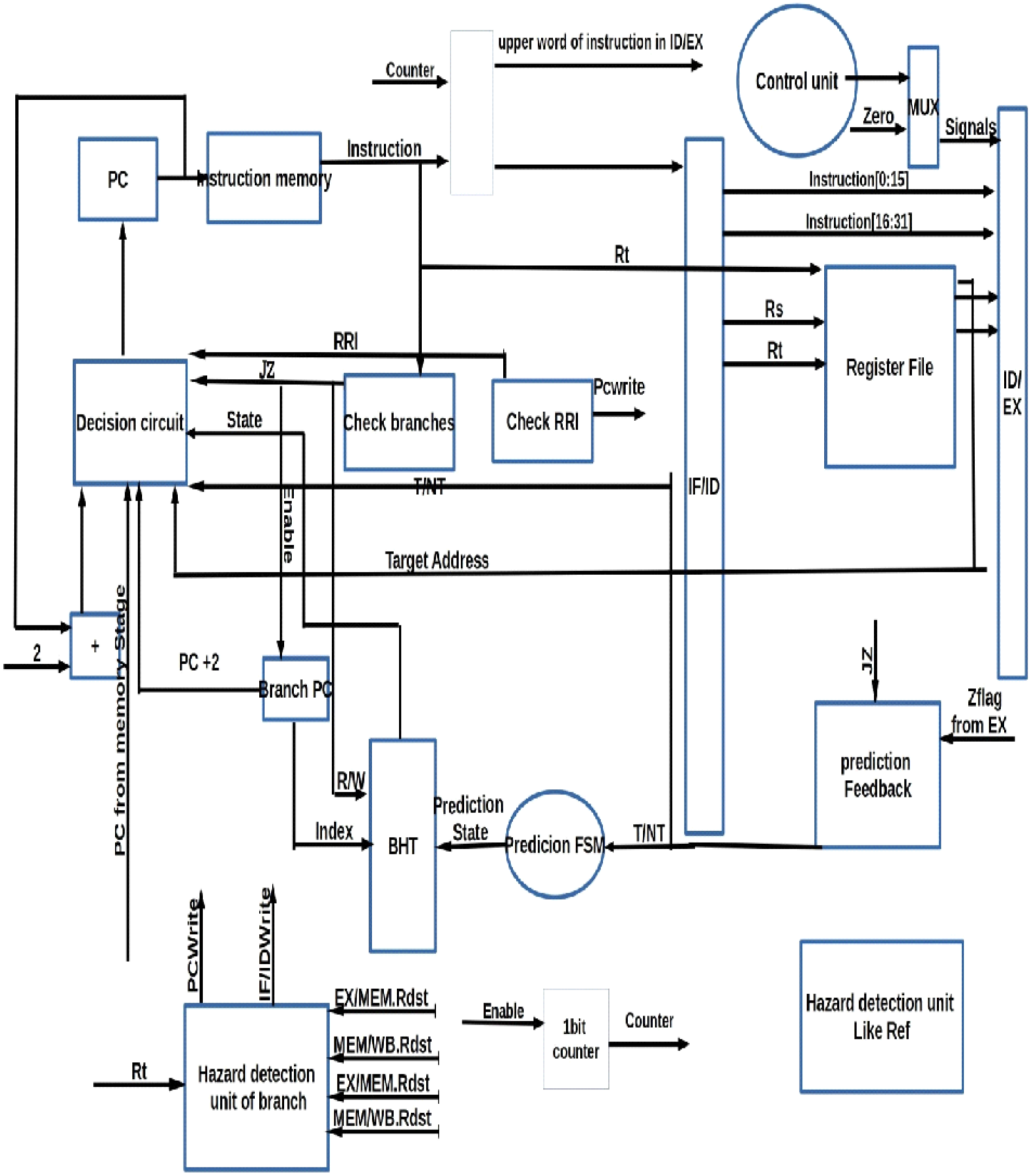
Opcode 5 bits	Rdst 3 bits	Rsrc 3bits	00000 5 bits

Opcodes for every Instruction

Instruction	Opcode
NOP	00000
NOT Rdst	00001
INC Rdst	00010
DEC Rdst	00011
OUT Rdst	00100
IN Rdst	00101
SWAP Rsrc, Rdst	00110
ADD Rscr1, Rscr2, Rdst	00111
IADD Rscr1, Rdst, Imm	01000
SUB Rscr1, Rscr2, Rdst	01001
AND Rscr1, Rscr2, Rdst	01010
OR Rscr1, Rscr2, Rdst	01011
SHL Rscr, Imm	01100
SHR Rsrc,Imm	01101
PUSH Rdst	01110
POP Rdst	01111
LDM Rdst, Imm	10000
LDD Rdst, EA	10001
STD Rscr, EA	10010
JZ Rdst	10011
JMP Rdst	10100
CALL Rdst	10101
RET	10110
RTI	10111

Schematic diagram:

Fetch and decode:



Notes:

There is a bus from instruction fetched to the second 16 bits of instruction section of ID/EX reg, to load the instruction fetched which is the address in case of 32 bit instruction and use it in Execution stage.

There is enable signals from prediction feedback and from check branches to enable the BHT to read or write, (check branches. JZ OR prediction Feedback. Enable) to disable the BHT in the ordinary instructions and enable it in case of JZ to read the state and the instruction next to JZ to write then new state.

Decision circuit: is used to determine the next pc value, to be loaded to the pc in falling edge after fetching the current instruction at the beginning of cycle (rising edge).

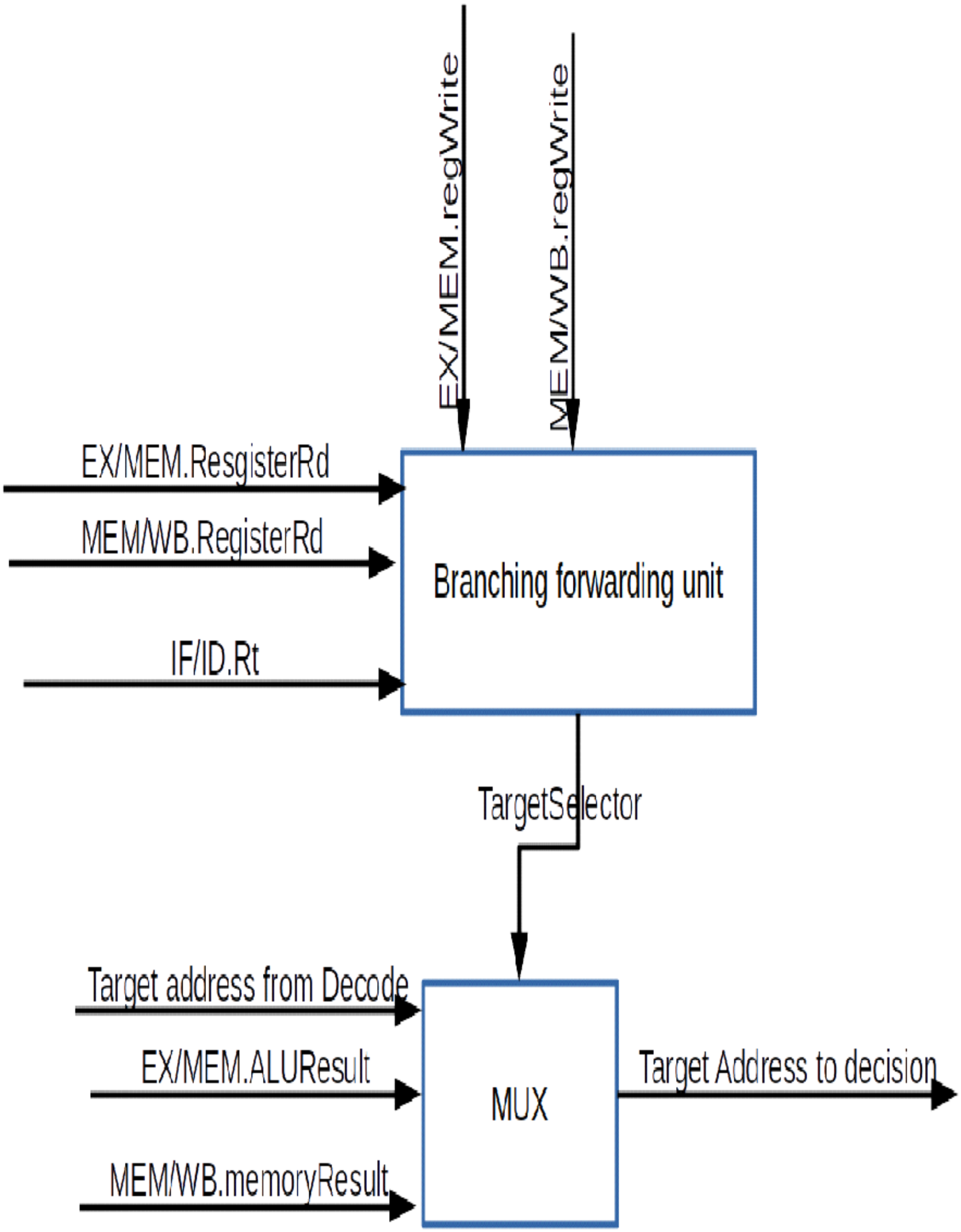
In case of branch instructions (jz, jmp, call, ret, RTI), at first we check for jz instruction which needs prediction, if the instruction is JZ then we load the current value of the pc to the branch pc.

Branch pc: used to save it to use in case of miss prediction (predict token and the prediction is wrong then we need to load the next instruction of branch instruction not the next of the token prediction target address).

BHT: used to save the state of JZ instructions, it has R/W input which determined by JZ signal.

Check RRI: to check for RET, RTI, interrupt instructions that need to write back the read PC from the memory stage, so it stalls the fetch stage.

Hazard Detection Unit of branch: needed to stall the fetch and decode in case of branch WAR data dependency (if.flush=1,pcWrite =0).


EX/MEM.regWrite
MEM/WB.regWrite
EX/MEM.ResgisterRd
MEM/WB.RegisterRd
IF/ID.Rt
Branching forwarding unit
TargetSelector
Target address from Decode
EX/MEM.ALUResult
MEM/WB.memoryResult
MUX
Target Address to decision

1bit Counter: used to determine if the current fetch stage is fetching address not instruction . 1 => address

- enable of counter is ORING of decode signal of 32/16 signal , 0th bit in instruction fetched.

decision circuit at IF/ID: its input is the fetched instruction and counter to determine the output which is zeros or the instruction. In case of counter =1 then the output will be the instruction forwarded to ID/EX is the instruction(address) and the another output is zeros to stored , if.flush = 1 ,,if counter =0 then the instruction stored in IF/ID and if.flush = 0.

IF/ID => interrupted 0 in normal flow.

1=> when stage receive interrupt signal it stall fetch make pcwrite =0, put interrupt signal =1 in IF/ID.

Other stages put interrupt signal in the next intermediate register =1, then the next cycle to this stage it will stall itself.

In memory we would operate 4 cycles:

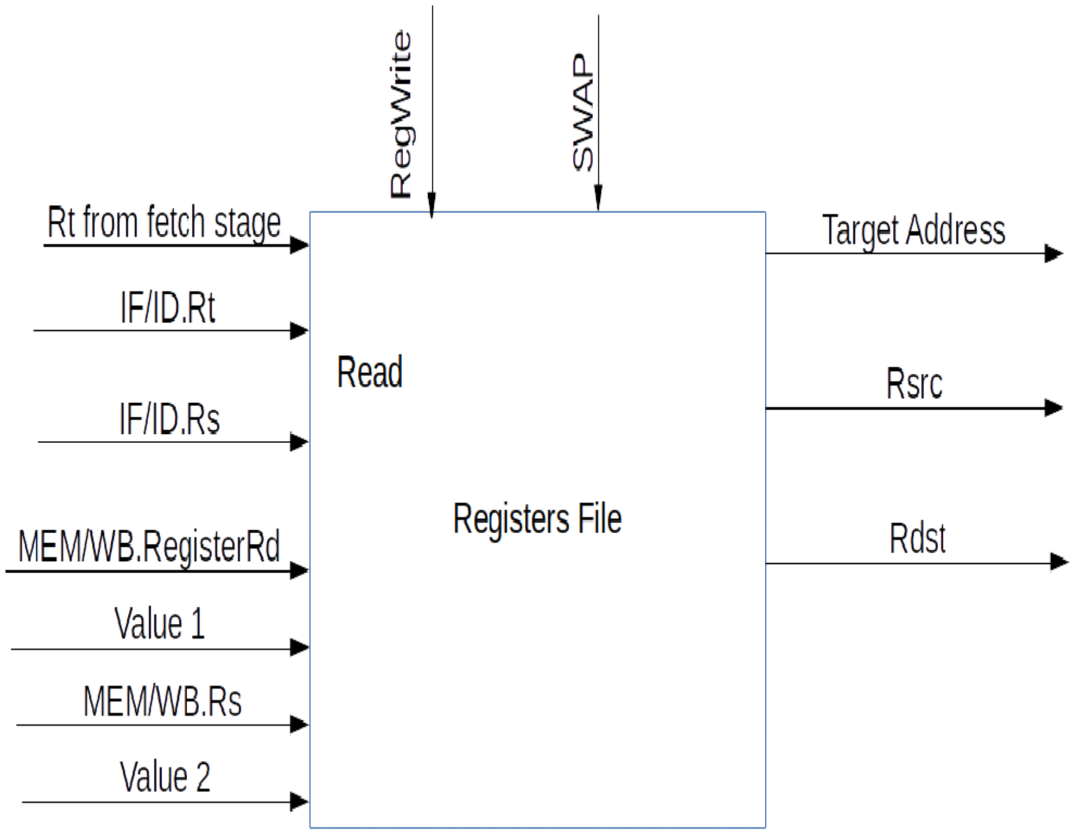
First cycle: to perform the current instruction.

Second cycle: to write PC value in the stack, after this cycle stops WB.

Third cycle: to put the address of M (2), M (3) to PC.

Fourth cycle: to store the flags in stack.

Decode:



at swap instruction : swap signal =1 then it enable the write functionality of write at Rs register.

Hazard detection unit

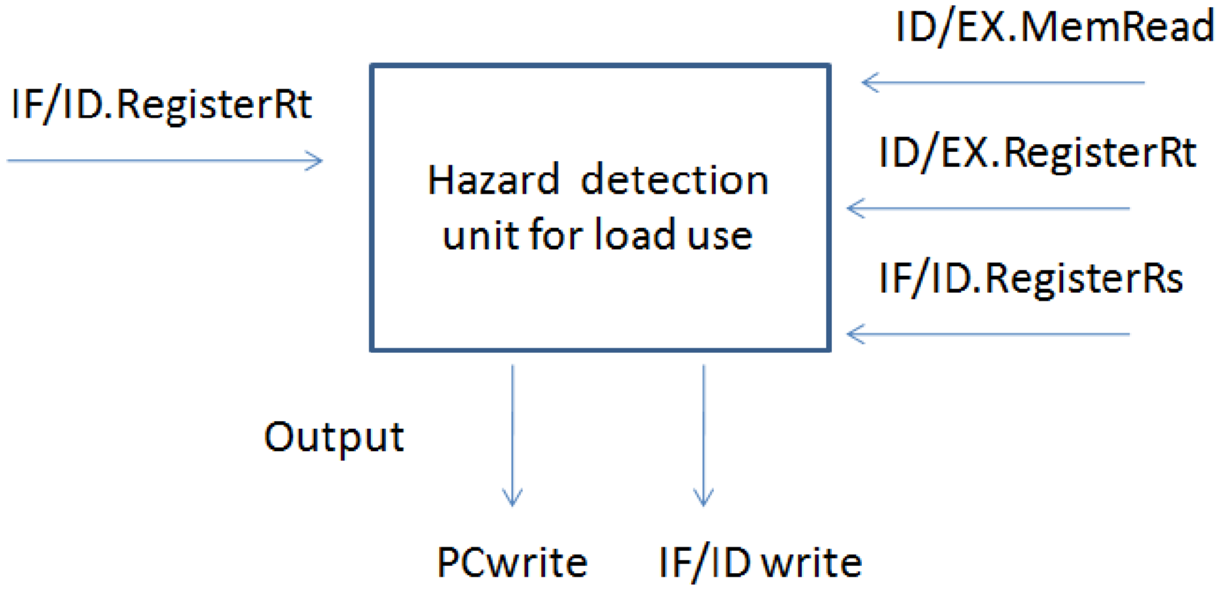
Checking for load instructions, the control for the hazard detection unit is this single condition:

if (ID/EX.MemRead and ((ID/EX.RegisterRt = IF/ID.RegisterRs) or

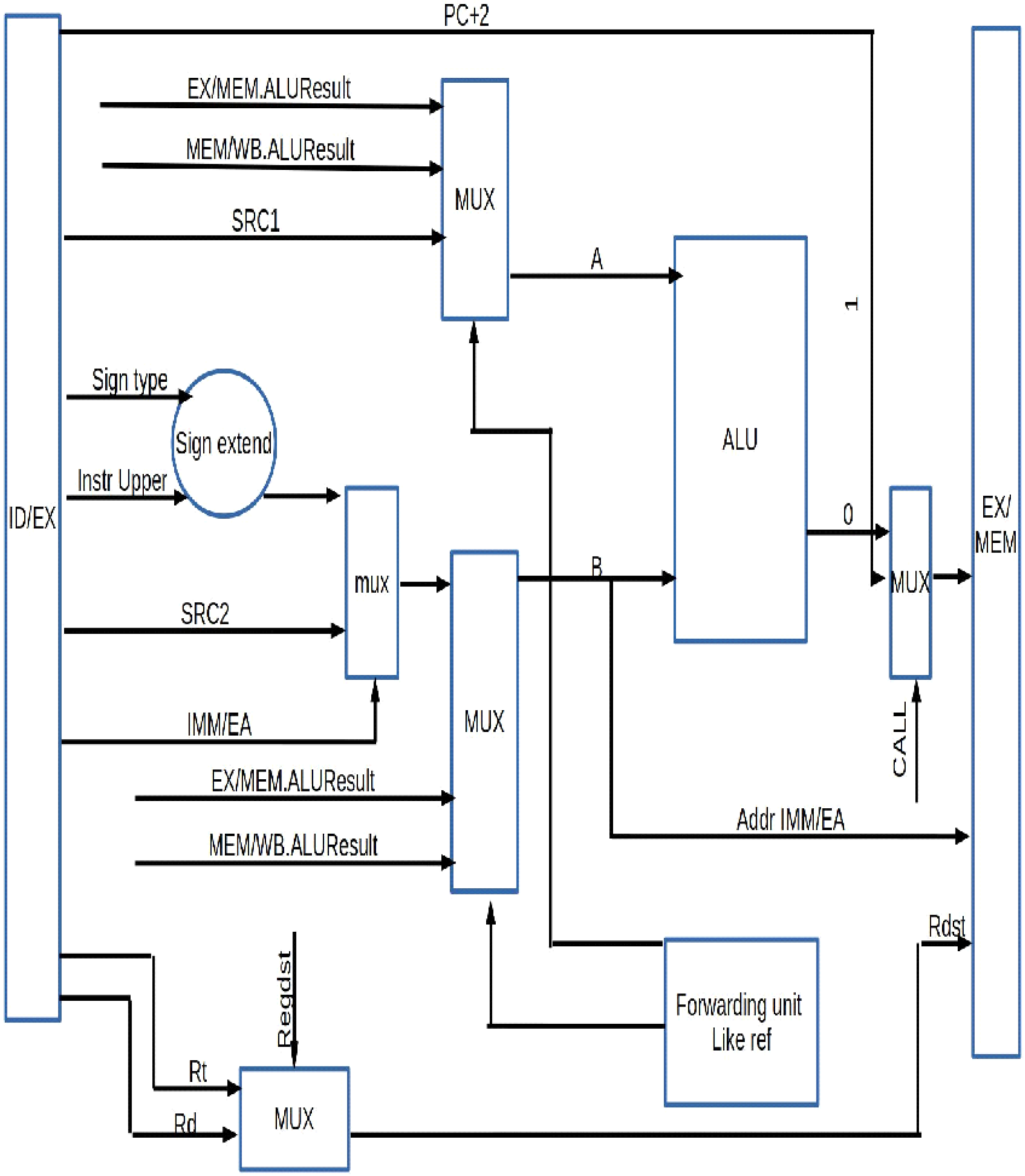
(ID/EX.RegisterRt = IF/ID.RegisterRt))   stall the pipeline

ID stage is stalled, then the instruction in the IF stage must also be stalled by preventing the PC register and the IF/ID pipeline register from changing. a stall bubble delays everything behind it.

The hazard detection unit controls the writing of the PC and IF/ID registers plus the multiplexor that chooses between the real control values and all 0s. The hazard detection unit stalls and deasserts the control fields if the load-use hazard test above is true.

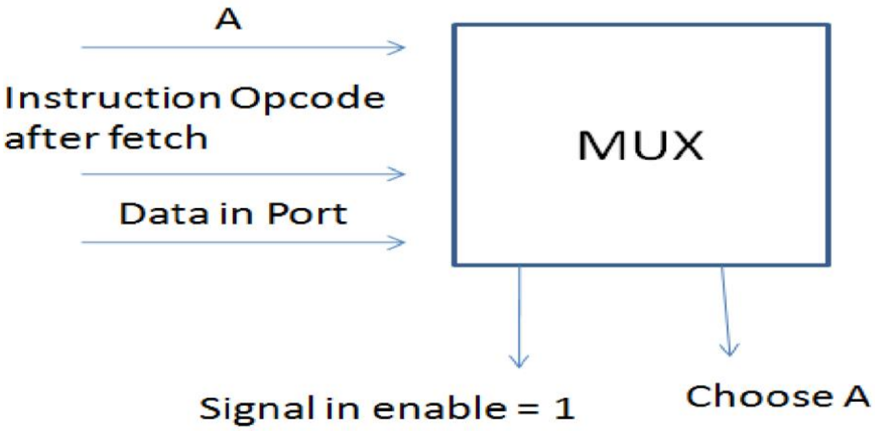

ID/EX.MemRead
IF/ID.RegisterRt
ID/EX.RegisterRt
IF/ID.RegisterRs
Hazard detection unit for load use
Output
PCwrite
IF/ID write

Execution :


PC+2
EX/MEM.ALUResult
MEM/WB.ALUResult
SRC1
MUX
A
ALU
1
Sign type
Sign extend
Instr Upper
ID/EX
mux
SRC2
B
MUX
0
MUX
EX/
MEM
IMM/EA
EX/MEM.ALUResult
MEM/WB.ALUResult
CALL
Addr IMM/EA
Rdst
Regdst
Forwarding unit
Like ref
Rt
Rd
MUX

- We add multiplexer before ALU takes A and input port value and selects between them.

  In this case selector will be one (signal **in enable =1).**



MUX to load to data to be written in the memory take the data from ALUResult , PC+1 from IF/ID register. The selector is Call signal which indicate that the instruction is CALL which need to write the pc value

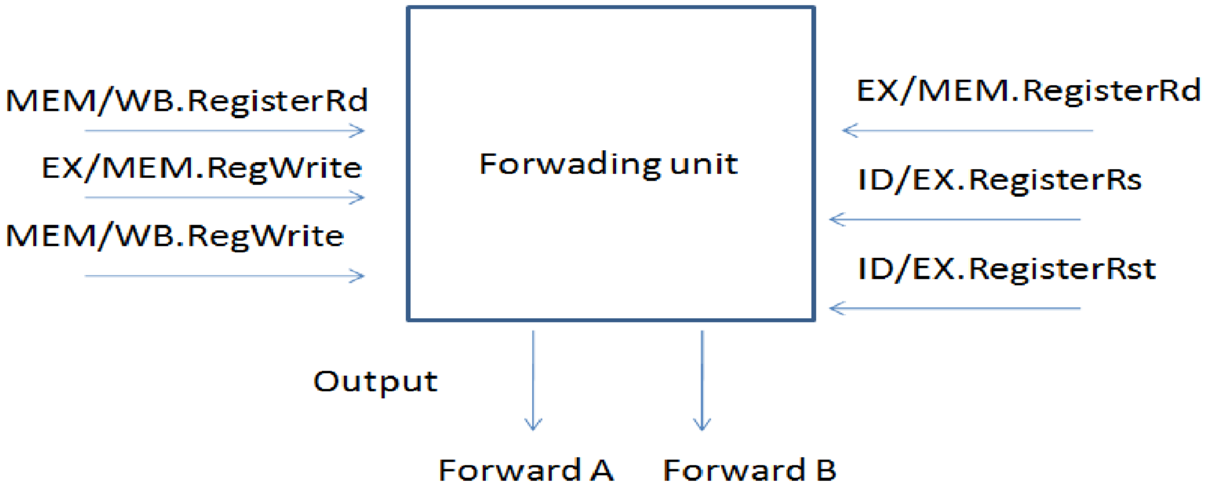
ALU selectors bits (4 bits):

- NOP: 0000
- A: 0001
- B: 0010
- INC:0011
- DEC:0100
- ADD:0101
- SUB:0110
- NOT: 0111
- AND: 1000

- OR: 1001
- SHL: 1010
- SHR: 1011

1- Data Hazard Forwarding Unit

Forwarding unit diagram



Data Hazard Conditions:

1a. EX/MEM.RegisterRd = ID/EX.RegisterRs

1b. EX/MEM.RegisterRd = ID/EX.RegisterRt

2a. MEM/WB.RegisterRd = ID/EX.RegisterRs

2b. MEM/WB.RegisterRd = ID/EX.RegisterRt

1. EX hazard:

If (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0) and

(EX/MEM.RegisterRd = ID/EX.RegisterRs)) ForwardA = 10

If (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0) and

EX/MEM.SWAP and

(EX/MEM.RegisterRs = ID/EX.RegisterRs)) ForwardA = 10

If (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0) and

(EX/MEM.RegisterRd = ID/EX.RegisterRt)) ForwardB = 10

If (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0) and

EX/MEM.SWAP and

(EX/MEM.RegisterRs = ID/EX.RegisterRt)) ForwardB = 10

2. MEM hazard:

If (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0) and

(MEM/WB.RegisterRd = ID/EX.RegisterRs)) ForwardA = 01

If (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0) and

MEM/WB.SWAP and

(MEM/WB.RegisterRs = ID/EX.RegisterRs)) ForwardA = 01

If (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0) and

(MEM/WB.RegisterRd = ID/EX.RegisterRt)) ForwardB = 01

If (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0) and

MEM/WB.SWAP and

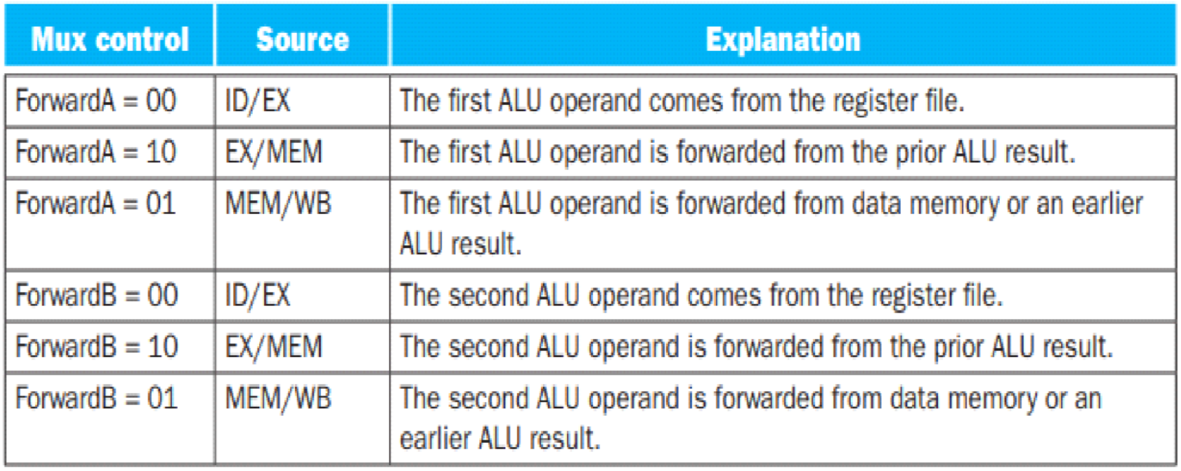
(MEM/WB.RegisterRs = ID/EX.RegisterRt)) ForwardB = 01

In this case, the result is forwarded from the MEM stage because the result in the MEM stage is the more recent result. The control for the MEM hazard would be :

if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0) and not(EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0) and

(EX/MEM.RegisterRd ≠ ID/EX.RegisterRs)) and (MEM/WB.RegisterRd = ID/EX.RegisterRs)) ForwardA = 01 if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0) and not(EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0) and (EX/MEM.RegisterRd ≠ ID/EX.RegisterRt))

and (MEM/WB.RegisterRd = ID/EX.RegisterRt)) ForwardB = 01



Mux control	Source	Explanation
ForwardA = 00	ID/EX	The first ALU operand comes from the register file.
ForwardA = 10	EX/MEM	The first ALU operand is forwarded from the prior ALU result.
ForwardA = 01	MEM/WB	The first ALU operand is forwarded from data memory or an earlier ALU result.
ForwardB = 00	ID/EX	The second ALU operand comes from the register file.
ForwardB = 10	EX/MEM	The second ALU operand is forwarded from the prior ALU result.
ForwardB = 01	MEM/WB	The second ALU operand is forwarded from data memory or an earlier ALU result.

Control hazards and branching :

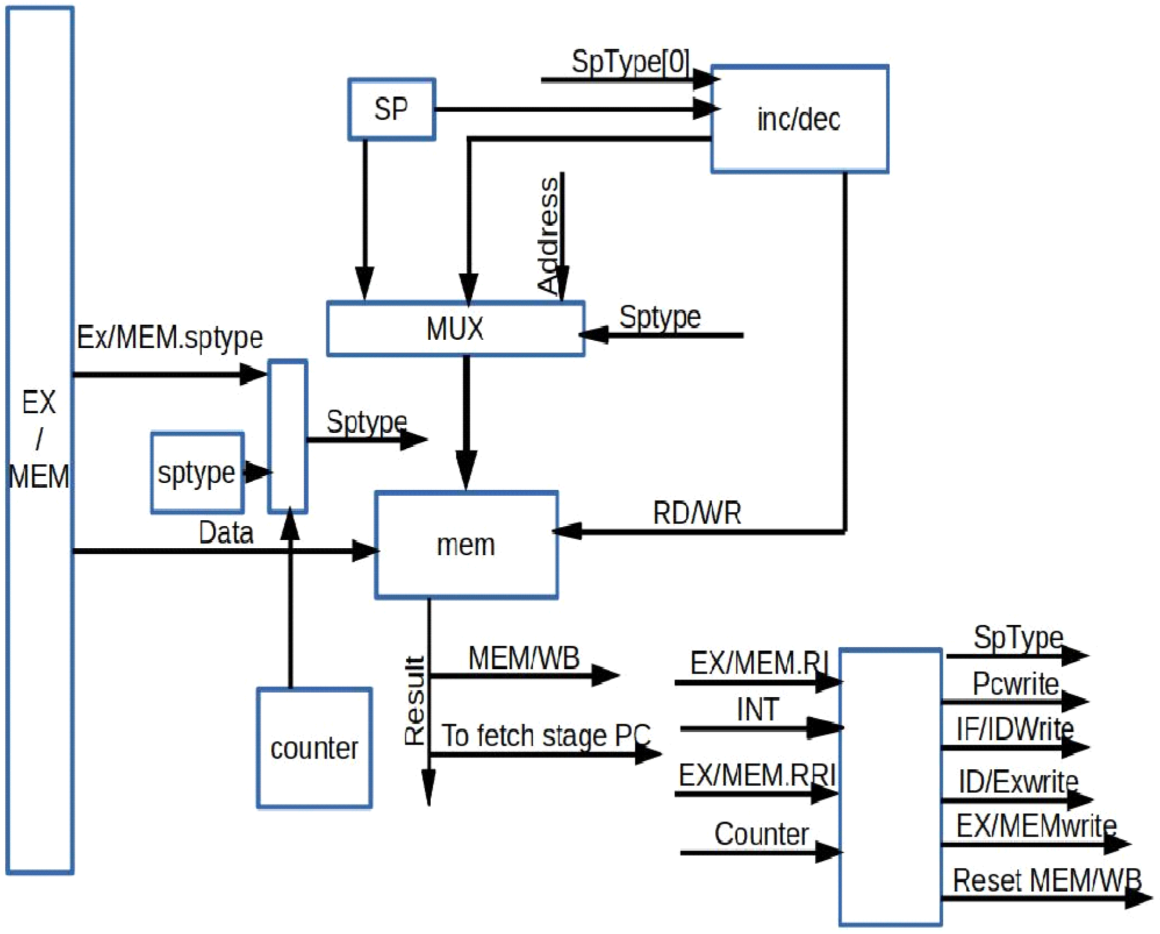
In fetch stage we add check branching and check RTI to determine the current instruction and check if it has control hazard or not ,

in case of unconditional branching JMP ,CALL, it gets the pc new value from decoding stage.

in case of JZ which need prediction at first it will read the prediction state from BHT then according to it it determine whether pc new will be PC+2(NT) or target address from decode,then in the next cycle in fetching of new instruction and decode of JZ instruction , then the prediction feedback circuit out the T/NT/notJZ to determined if the prediction was right or not ,and to determine the new state to store in BHT.

RET,RTI : they need the pc from memory stage then in each stage if RRI =1 then it will stall itself and the previous stage to prevent any new instruction from execution , and in memory stage it will stall itself and the previous stages and determine the logic of memory according to the instruction.

Mem



- Counter enable determined by RRI, RI, interrupt signal.

- There is a Decision circuit to determine data input to memory.
- Inputs are counter, RI, RRI, interrupt signal to be selectors, and the data to be selected pc from fetch, flags from ALU, EX/MEM.Data
- There is a Decision circuit to determine address of selector.
- Inputs are counter ,RI,RRI,interrupt signal, reset signal to be selectors, and the addresses to be selected 0 for reset ,2 for interrupt, EX/MEM.address.

Pipeline registers details:

IF/ID register

Inputs of this register are IR (2 bytes) + PC new (after incremented) (4 bytes), interrupt signal (1 bit), RET(1 bit),RTI(1 bit), reset (1 bit),INPORTValue(4 bytes). Instruction of fetch stage is navigated to ID/IE to be stored in the lower 16 bits of the register. (84 bits)

In case of data hazards:

- PC needs an enable in case of stall the pipeline.
- IF/ID has an enable so that it will not be changed.

In case of control hazard

In jz instruction when decode decision is opposite to prediction then prediction FSM make make if.flush = 1

ID/EX:

Needs the values are read from the register file of Rscr1 and Rscr2 so 4 Bytes for Rscr1 + 4 Bytes for Rscr2, instruction (4 bytes),PC(after increment)(4 bytes),Ret(1 bit),RTI(1 bit),SWAP, CALL,INT

- WBsignals = 3 signals => memtoreg, regwrite, outenable =>3bits.
- EXsignals =ALUop=>4bits.
- SignExtend, IMM/EA, regDst, INEnableSignal=>4bits.
- MEM signals = memRead, memWrite, spType=>4bits.
- InPortValue(4Bytes)

(180 bits)

EX/MEM:

It must be big enough to hold all possible situations as the following:

- Rsrc2,SWAP,Rt
- ALUresult/pc+1 =>4bytes, address =>4bytes, Rdst (3 bit).
- Interrupt signal (1 bit), RET(1 bit),RTI(1 bit),CALL(1 bit),CRR (flags) (3 bits).
- MEM signals = memRead, memWrite, spType=>4bits.
- WBsignals = 3 signals => memtoreg, regwrite, outenable =>3bits.

(117 bits)

MEM/WB:

- Rsrc2(4bytes),SWAP,Rt,Rd
- WBsignals 3bits
- Result from the memory (.i.e. in load instruction it will result 4bytes for the value to be WB in DST reg).
- Result from ALU (4bytes).

(106 bits)

Fetching 2 word instructions:

Bit 1 in instruction is indicator to the type of instruction:

- ‘0’ if the instruction is 16 bits.
- ‘1’ if the instruction is 32 bits.

In fetch stage we know the instruction then use the 1th bit of instruction to enable the counter.

When counter =0 then the stage fetch the first 16 bits of the instruction , then it make decision circuit will store the instruction in IF/ID , and forward zeros to ID/EX upper word of instruction section ,then in the next cycle we decode the first part of instruction and we know that the instruction is 32 bits then 32/16 =1 then counter will be 1 and when 32/16 =1 then if.flush = 1 to insert bubble to the remaining of the fetched instruction which is address. When counter =1 then instruction forwarded to ID/EX instruction section.

Control signal of each instruction:

One operand:

instr\ signals	RegWr	RegDST	Mem To Reg	Mem Rd	MemWR	SP	ALU	PC write	Imm / EA	sign	CRR	in enable	Out enable
NOP	0	0	0	0	0	11	nop	1	0	0	0	0	0
NOT Rdst	1	1	0	0	0	11	A	1	0	0	0	0	0
INC Rdst	1	1	0	0	0	11	A+1	1	0	0	0	0	0
DEC Rdst	1	1	0	0	0	11	A-1	1	0	0	0	0	0
OUT Rdst	0	1	0	0	0	11	nop	1	0	0	0	0	1
IN Rdst	1	1	0	0	0	11	A	1	0	0	0	1	0

Branch:

instr\ signals	Reg WR	Reg DST	Mem To Reg	out enable	Mem Rd	mem WR	SP	Al U	32 /16	Imm/ EA	sign	CRR
PUSH	0	x	0	0	0	1	00	A	0	x	1	0
POP	1	1	1	0	1	0	01	nop	0	0	1	0
LDM	1	1	0	0	0	0	11	B	1	1	1	0
LDD	1	1	1	0	1	0	10	nop	1	1	0	0
STD	0	1	0	0	0	1	10	A	1	1	0	0
JZ	0	0	0	0	0	0	11	nop	0	0	x	0
JMP	0	0	0	0	0	0	11	nop	0	0	x	0
CALL	0	0	0	0	0	1	00	nop	0	0	x	1
RET	0	x	0	0	1	0	01	nop	0	0	x	1

RTI	0	x	0	0	1	0	01	nop	0	0	x	1
instr\signals	Reg WR	Reg DST	Mem To Reg	out enable	Mem Rd	mem WR	SP	ALU	32/16	Imm/EA	sign	CRR
Swap	1	1	0	0	0	0	11	A	0		0	0
ADD	1	1	0	0	0	0	11	Add	0	0	0	0
IADD	1	1	0	0	0	0	11	Add	0	1	1	0
SUB	1	1	0	0	0	0	11	Sub	0	0	0	0
AND	1	1	0	0	0	0	11	And	0	0	0	0
OR	1	1	0	0	0	0	1111	or	0	0	0	0
SHL	1	0	0	0	0	0	1111	shl	1	1	0	0
SHR	1	0	0	0	0	0	1111	shr	1	1	0	0