Name: Duc Anh Nguyen

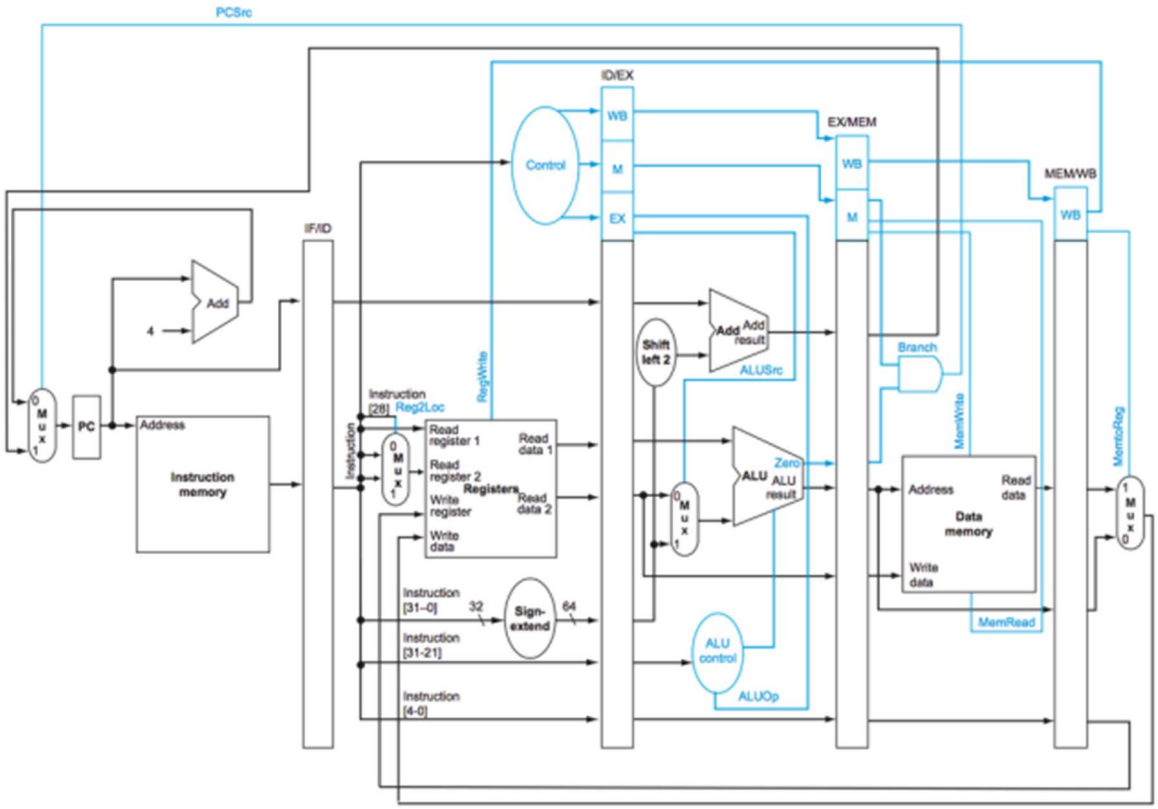
Course: EE126

Date: 10.24.2022

Lab Report 3

I. Introduction:

- The goal of this lab is to implement the pipeline LEGv8 CPU in this figure without a hazard detection or data-forwarding unit. Moreover, it used the previous implementation of the single-cycle CPU from the previous lab.



II. Procedure:

- These are the instructions that is going to be implemented in this lab

```
ADD X11, X9, X10        10001011000010100000000100101011
STUR X11, XZR,0         11111000000000000000001111101011
SUB X12, X9, X10        11001011000010100000000100101100
STUR  X11, [XZR,0]      11111000000000000000001111101011
STUR  X12, [X12,8]      11111000000000001000000110001100
STUR  X12, [X12,8]      11111000000000001000000110001100
ORR X21, X19, X20       10101010000101000000001001110101
NOP
NOP
STUR X21, [XZR,0]       11111000000000000000001111110101
NOP
NOP
NOP
NOP
LSR X21, X19, X20       11010011010101000000001001110101
```

- The contents of the register and memory:

Registers

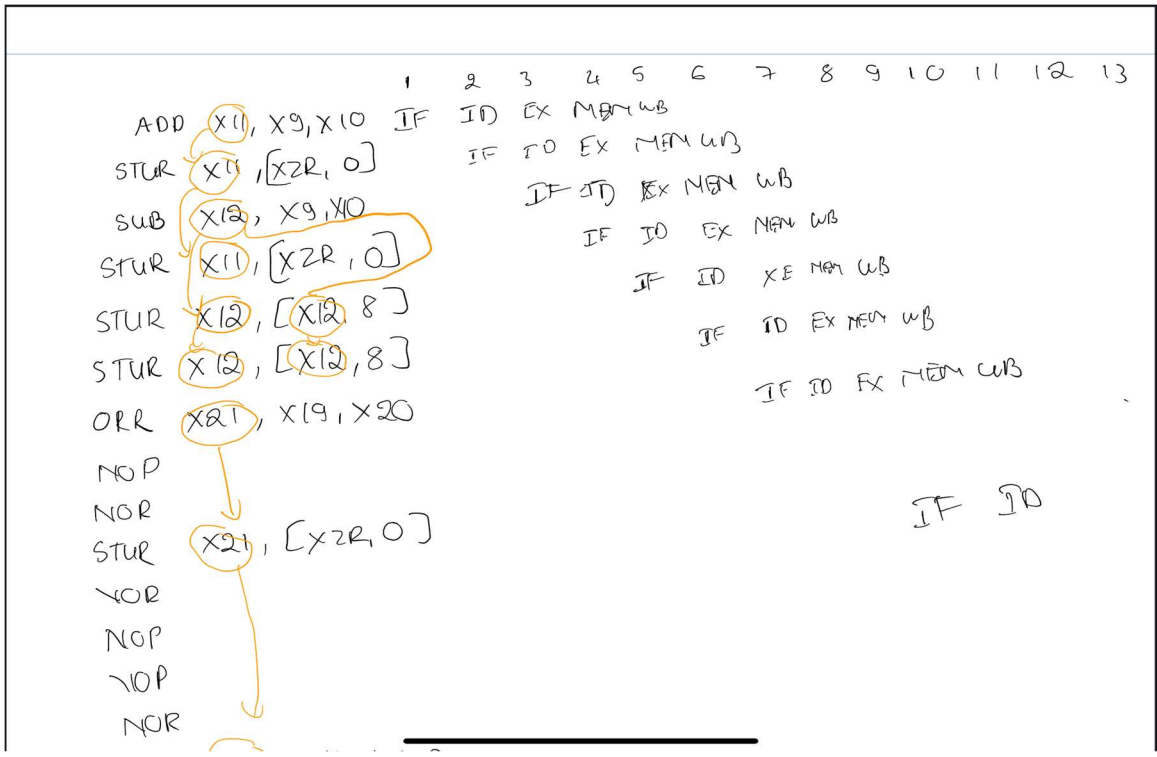
```
$X9  - 0x0000000000000010
$X10 - 0x0000000000000008
$X11 - 0x0000000000000002
$X12 - 0x000000000000000A
$X19 - 0x00000000CEA4126C
$X20 - 0x000000001009AC83
$X21 - 0x0000000000000000
$X22 - 0x0000000000000000
```

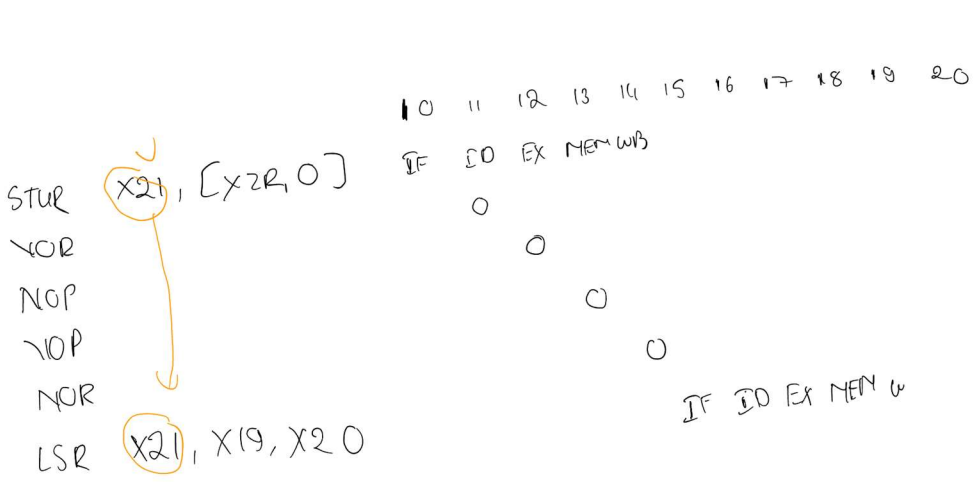
DMEM

```
DMEM(0x0)  - 1
DMEM(0x8)  - 2
DMEM(0x16) - 3
DMEM(0x24) - 4
```

DMEM values are in Hex.

- To understand the pipeline during each cycle, I have drawn the cycles of these instructions to understand how the pipeline registers communicate with each other.


1 2 3 4 5 6 7 8 9 10 11 12 13
ADD X11, X9, X10 IF ID EX MEM WB
STUR X11, [X2R, 0] IF ID EX MEM WB
SUB X12, X9, X10 IF ID EX MEM WB
STUR X11, [X2R, 0] IF ID EX MEM WB
STUR X12, [X12, 8] IF ID EX MEM WB
STUR X12, [X12, 8] IF ID EX MEM WB
ORR X21, X19, X20 IF ID EX MEM WB
NOP
NOR
STUR X21, [X2R, 0]
XOR
NOP
NOP
NOR
IF ID

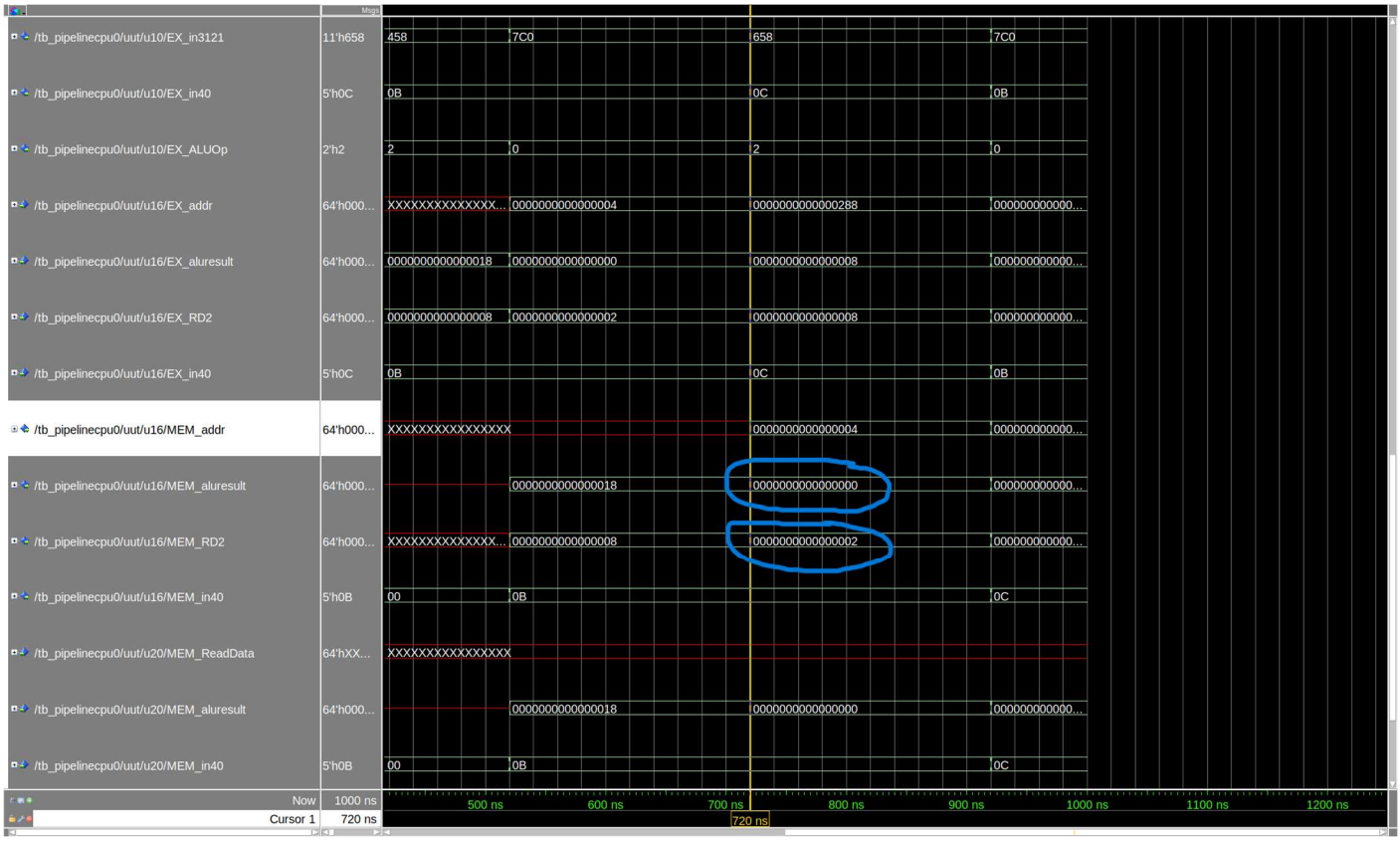

10 11 12 13 14 15 16 17 18 19 20
STUR X21, [X2R, 0] IF ID EX MEM WB
XOR O
NOP O
NOP O
NOR
LSR X21, X19, X20
IF ID EX MEM W

- In the upper figure, the data dependencies are colored in orange and the arrow will point out next instructions that is going to be affected by the data dependencies.

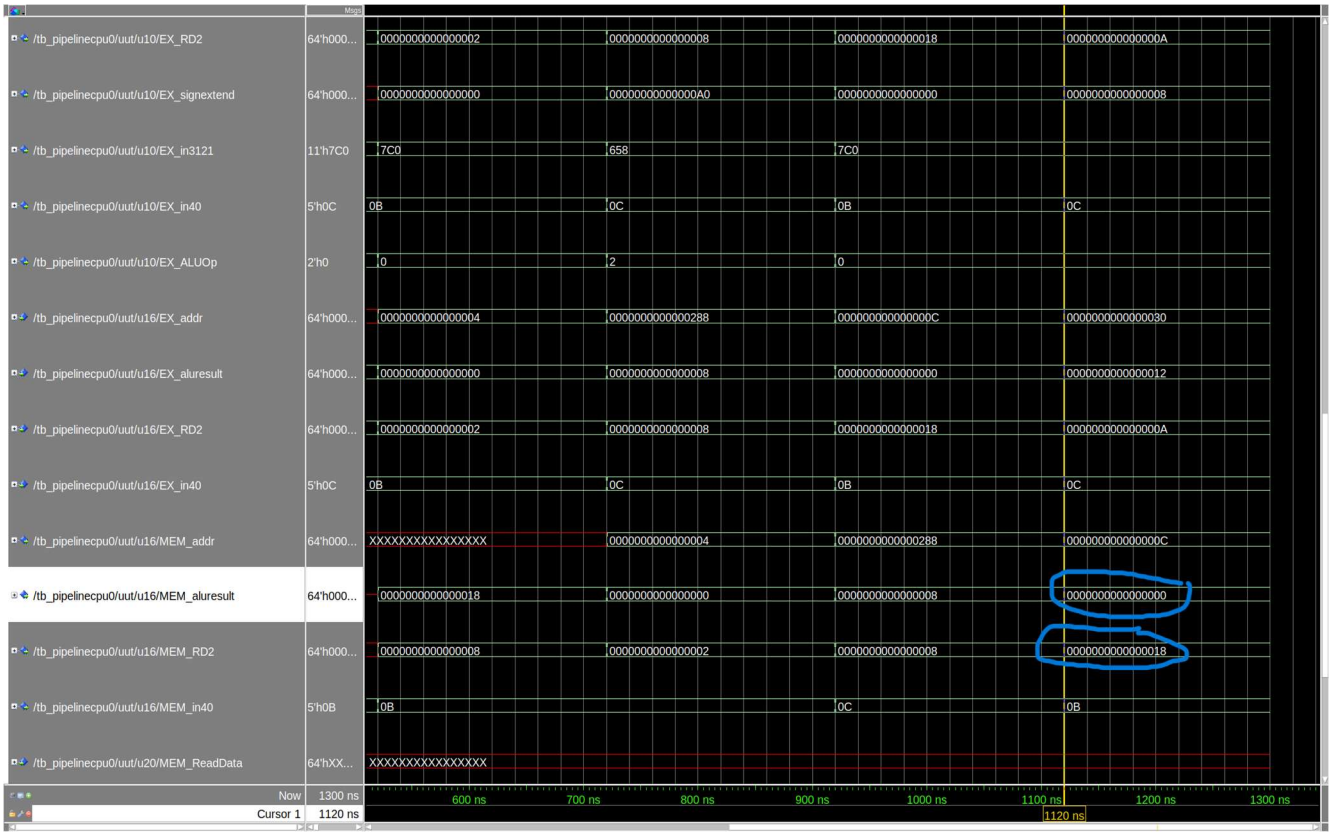
- Let's analyze the first data hazard with the register X11 in instruction 1, 2, and 3.

- In instruction 1, X11 is going to be the destination source of X9 and X10. However, the results are not going to be written until cycle 5 in the WB stage, which results in instructions getting the wrong value of X11 in the next instruction.

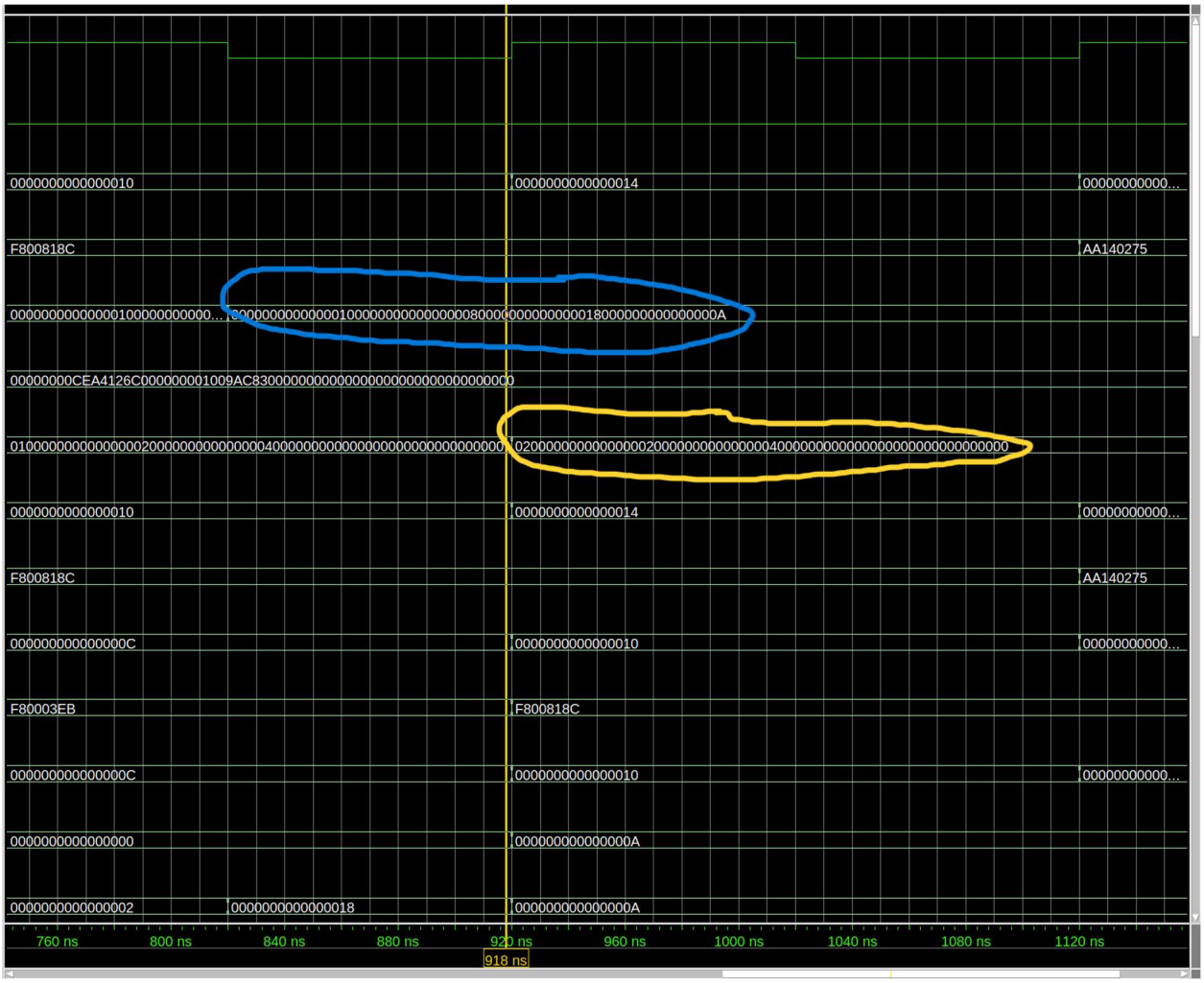
- The results of X11 of instruction 1 should be 18 as X10 + X9 = 18



- In the above figure, in the MEM stage of instruction 2 at cycle 5, the address, MEM_aluresult, and the write data, in which MEM_RD2, is not what we expected it to be as the write data should be the new value of X11, which is 18.

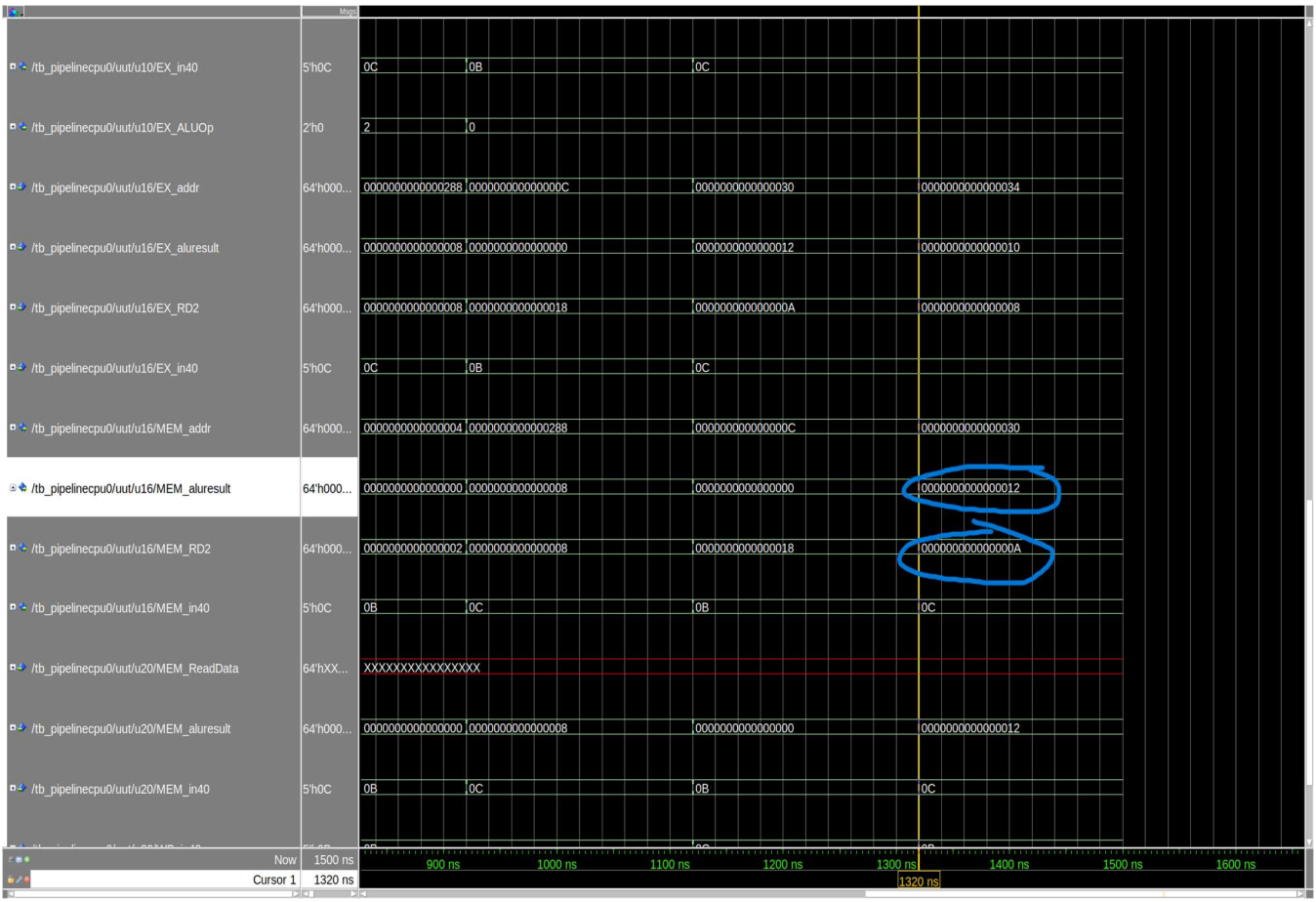


- Now in instruction 4, during the EX-stage at cycle 6 and MEM stage at cycle 7, the value of X11 has passed through the WB stage at cycle 5. Therefore, the new value of the instruction load into address should be correct

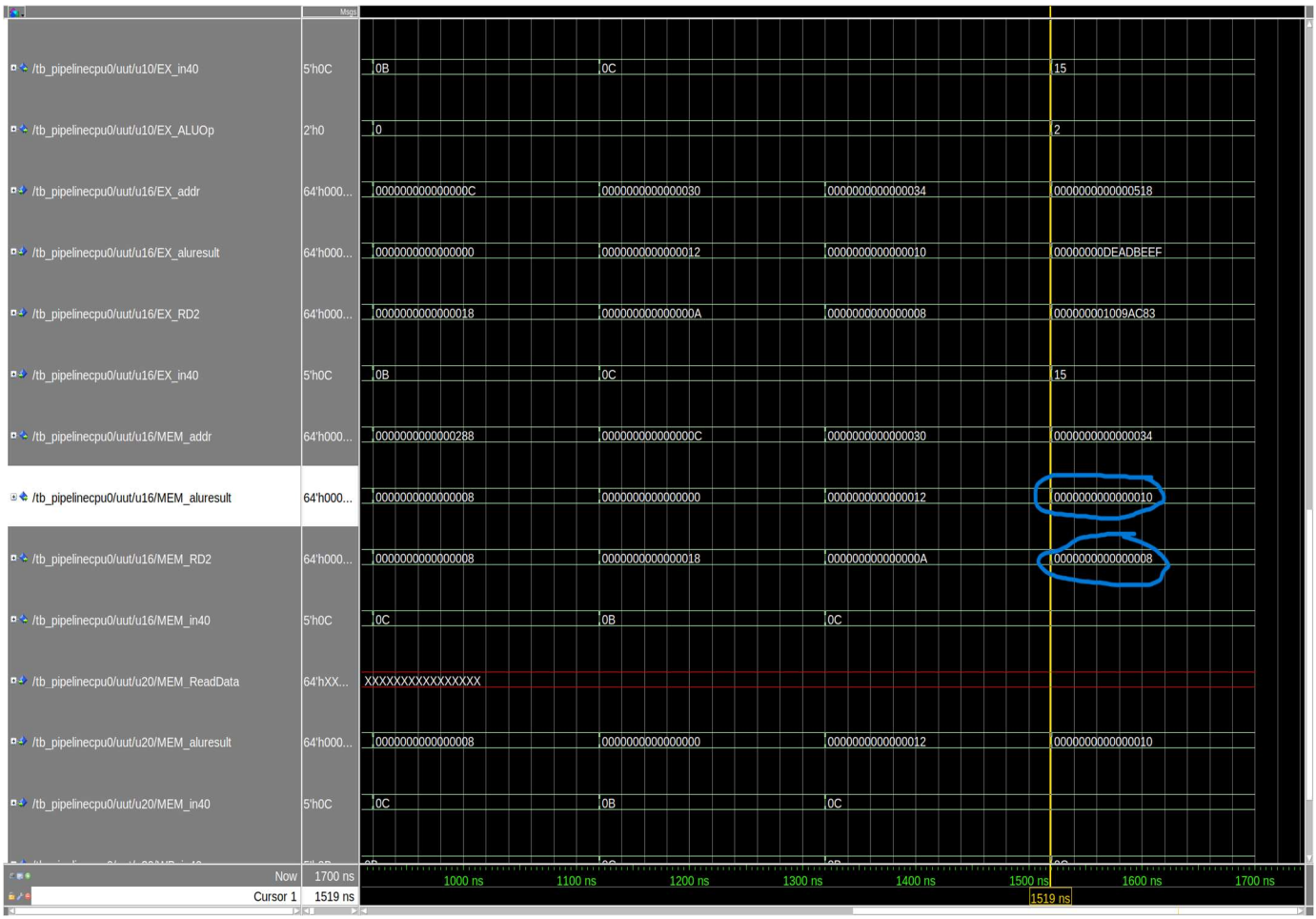


- The figure above shows the values of register and memory contents at cycle 5 and 6. The blue markup is the value of the registers X11 being updated during the WB stage of cycle 5, and the yellow markup is the value of memory contents being updated using the old values of X11, which is Hx002, that we have discussed above.

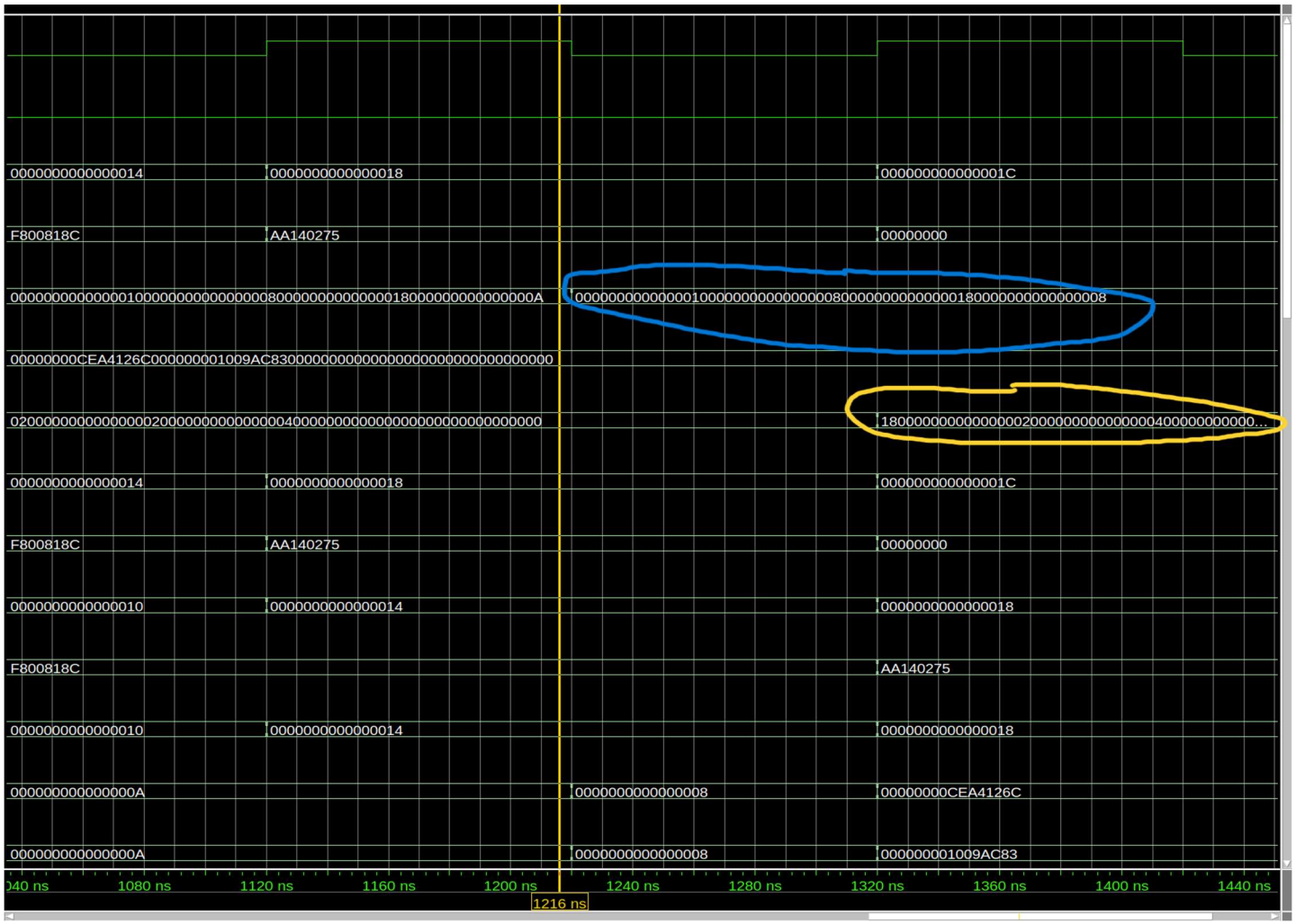
- The next data hazard is going the dependencies of X12 in instruction 3, 5, and 6, which is like the problem of X11 dependencies.
- The value of new X12 is X9 – X10 = 10 – 8 = 8 (hexadecimal)
- As instruction 4 depends on the instruction 3 to finish writing into register X12 during cycle 7, it fetches the old value of X12 in the EX-stage and MEM stage, illustrated by the blue markup of the figure below.



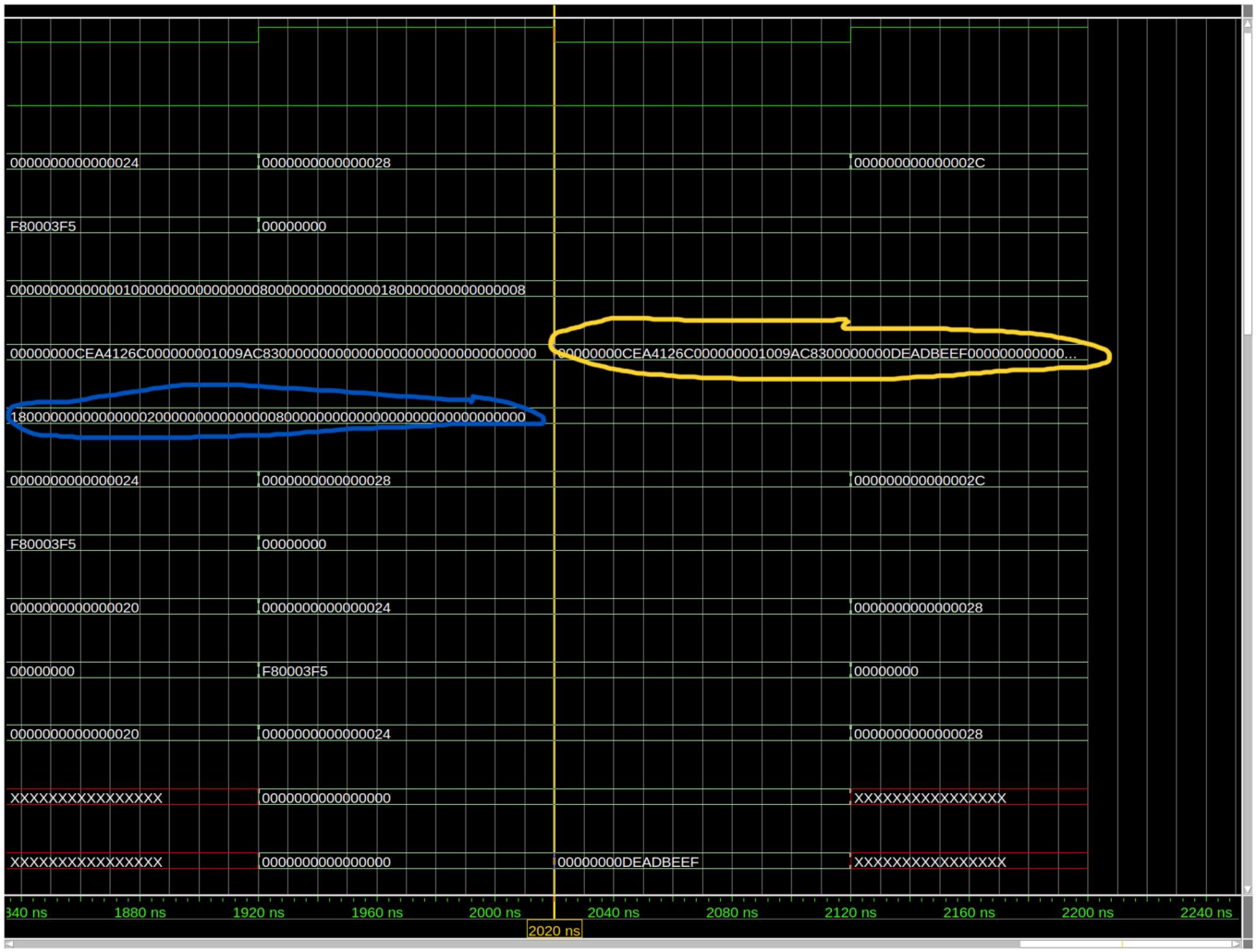
- In instruction 6, the EX and MEM stage fetch the new values of X12 as instruction 3 finishes its WB stage. We can see those values below.



- These are cycle 7 and 8 of instruction 3 and 4 in the figure below. The blue markup is the changes to X12 by instruction 3 with its new value, and the yellow markup is the changes to the memory contents with the new value of X11 for instruction 4

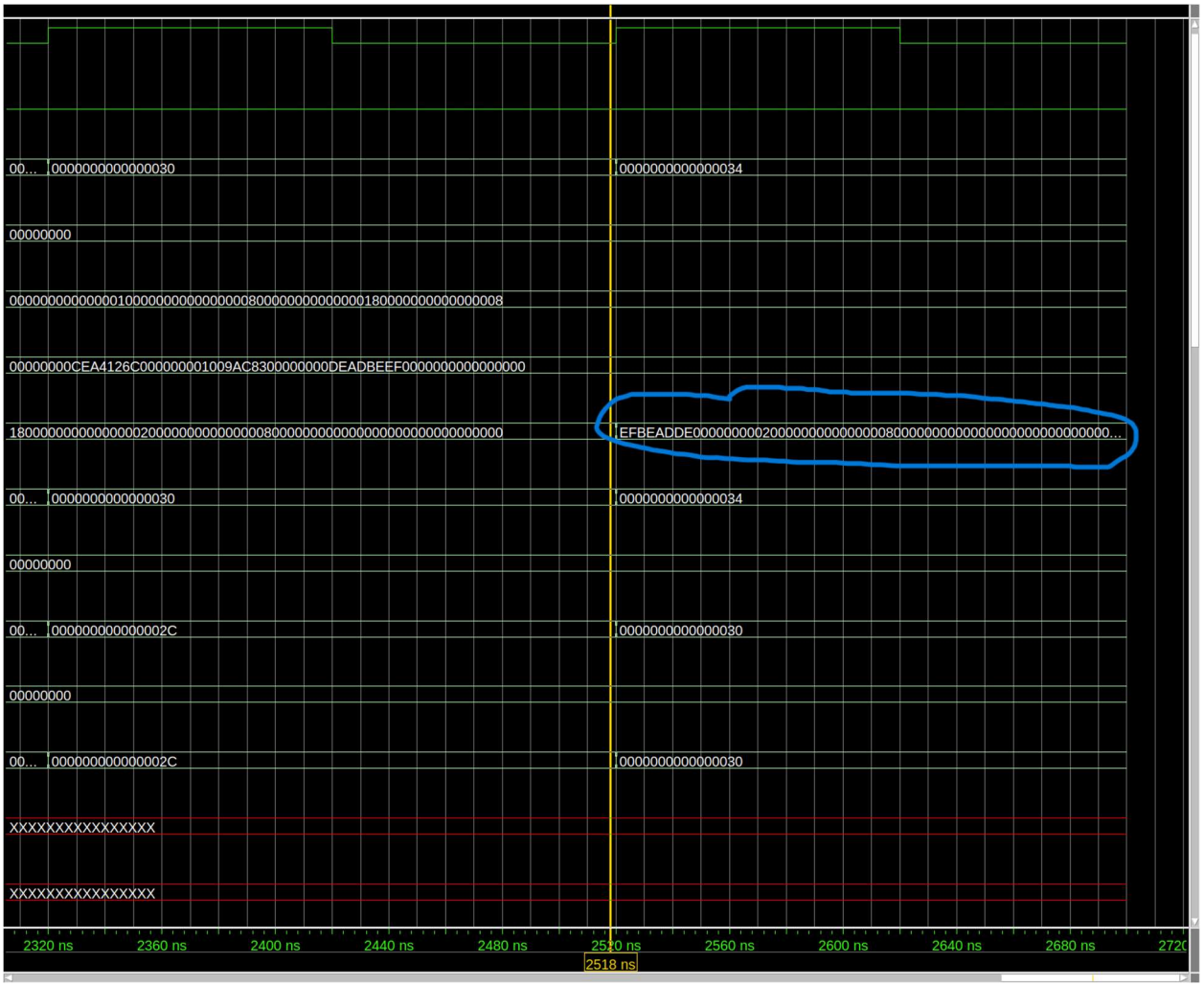


- The next data dependency is register X21 for instruction 7, 10, and 17, but because NOP has been applied to prevent data hazard, there should not be any wrong value being stored.

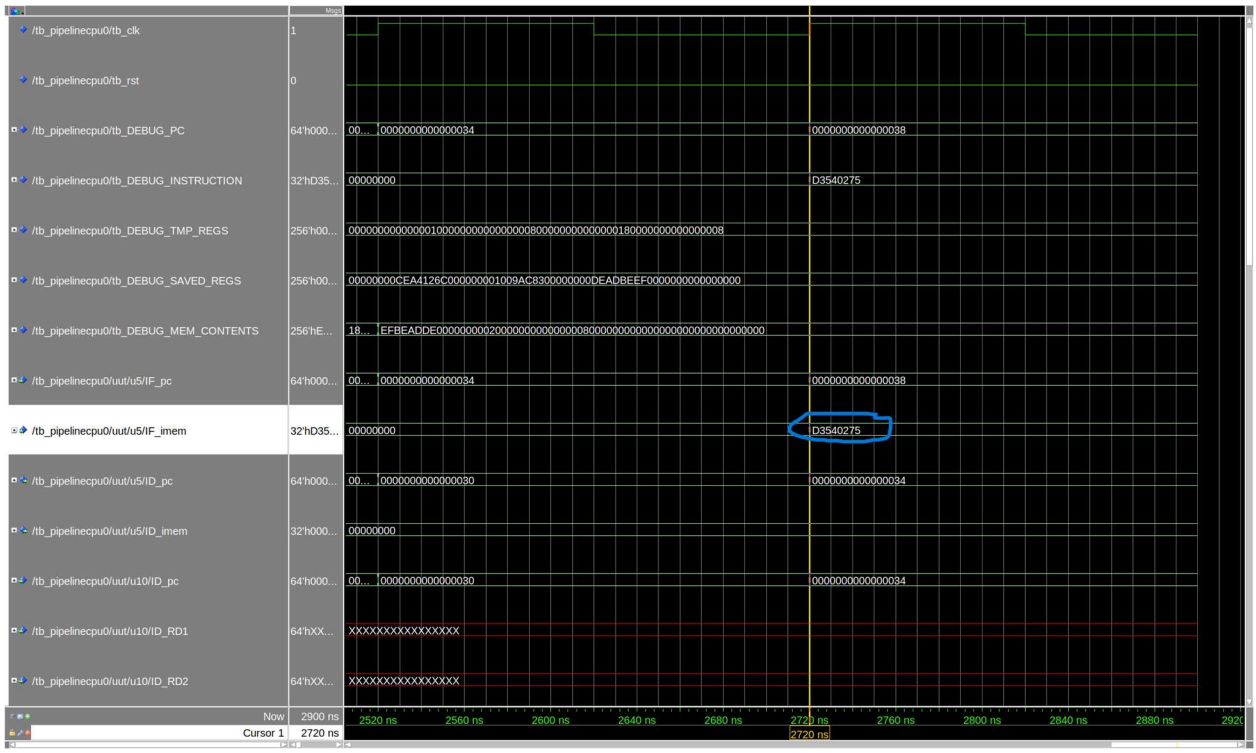


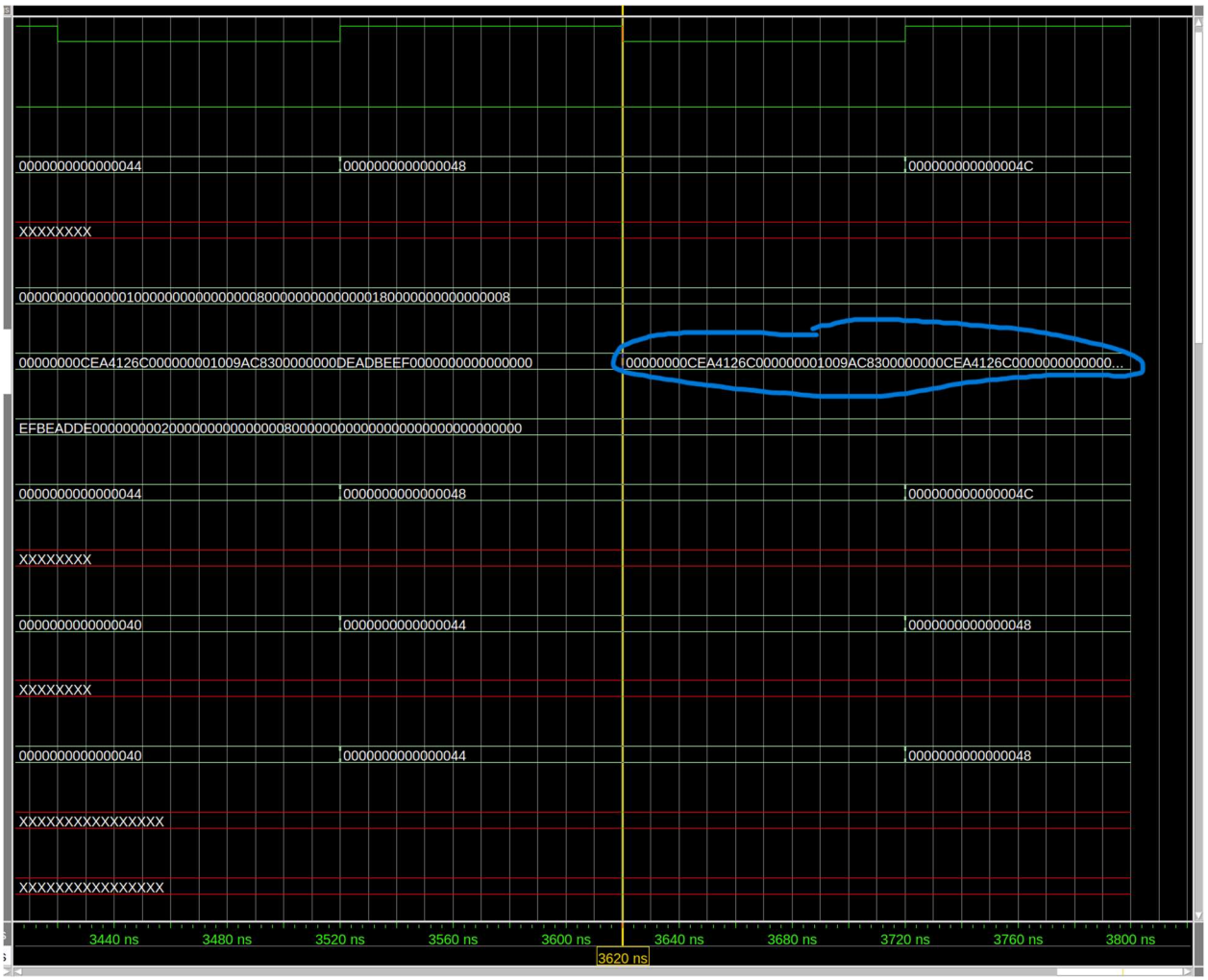
- During cycle 10 and 11, the results of instruction 6 and 7 has been shown in the above figure. The yellow markup is the results of instruction 7 and the results are stored in X21, and the blue markup is the new value of X12, which is 8, being stored in memory location Hx0010.

- In the figure below are the results of instruction 10, with the blue markup indicates the results.



- The figure below shows the moments where the CPU fetches instruction 15 and produce the results at cycle 19.




0000000000000044
0000000000000048
000000000000004C
XXXXXXXX
00000000000000100000000000000008000000000000001800000000000000008
00000000CEA4126C000000001009AC8300000000DEADBEEF0000000000000000
000000000CEA4126C000000001009AC8300000000CEA4126C0000000000000...
EFBEADDE0000000002000000000000000800000000000000000000000000000000
0000000000000044
0000000000000048
000000000000004C
XXXXXXXX
0000000000000040
0000000000000044
0000000000000048
XXXXXXXX
0000000000000040
0000000000000044
0000000000000048
XXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXX
3440 ns
3480 ns
3520 ns
3560 ns
3600 ns
3620 ns
3640 ns
3680 ns
3720 ns
3760 ns
3800 ns