Cycle: operation

1: load 0 into register 5

2: load 2 into register 6

3: load 2 into register 7

4: load 2 into register 8

5: load 2 into register 9

6: load 2 into register 18

7: load 2 into register 19

8: load 2 into register 20

9: load 2 into register 21

10: load 2 into register 22

11: load 2 into register 23

12: load 2 into register 24

13: load 2 into register 25

14: load 2 into register 26

15: load 2 into register 27

16: load 2 into register 28

17: load 2 into register 29

18: load 2 into register 30

19: load 2 into register 31

20: add register 24 with immediate store in register8

10+ 10101 = 10111

21: logical left shift register 5 by immediate store in register31

Shift 10 by 11101 = 1000000000000000000000000000000

22: xor register 20 with immediate store in register20

00010

11011 =

11001

23: shift logical right register 18 by immediate store in register5

10 by 110 = 0

24: shift arithmetic right register24 by immediate store in register19

10 by 10000001000 = 0

25: or register20 with immediate store in register26

10

11100 =

11110

Here we need forwarding so will be fixed in the next datapath

26: branch if register7 equal to register31

Sub 10 and 10

0

Equal BRANCH -> jump 2 addresses

STALL for three cycles

All zeros

Not equal NO BRANCH

27: Brach if register21 equal to register28

Sub 10 and 10 = 0

28: and register30 with immediate store in register21

00010

11001 =

00000

28: add register 22 with register 20 store in register24

10 +11001 = 11011

29: subtract register25 with register 8 store in register21

10 – 10111 = 1111111111111111111111111111111111111111111111111111111111101011

30: logical left shift register 9 by register 28 store in register18

10 by 10 = 1000

31. Branch if register18 not equal to register9

Sub 10 and 10 = 00

With carry

EQUAL NO BRANCH

STALL for three cycles

All zeros

33: branch if register5 not equal to register6

Sub 0 and 10 1111111111111111111111111111111111111111111111111111111111111110

Not equal BRANCH -> jump 4 addresses

STALL for three cycles

All zeros

34: xor register 7 with register7 store in register8

10

10 =

0

35: shift logical right register25 by register5 store in register21

10 shift by 0 = 10

32: shift arithmetic right register22 by register30 store in register19

10 by 10 = 0

33: or register20 with register6 store in register5

11001

10 =

11011

34: and register30 with register29 store in register21

10

10 =

10

35: branch if register18 is less than register9

1000 sub 10 = 0110

Carry

Not less than NO BRANCH

STALL for three cycles

All zeros

36: branch if register 7 is less than register31

Sub 10 and 000000000000000000000000000000001000000000000000000000000000000

1111111111111111111111111111111111000000000000000000000000000010

No carry

Less than BRANCH -> jump 6 addresses

STALL for three cycles

All zeros

39: Store what is in register23 in data\_mem address 6

Store 10

40: store what is in register9 in data\_mem address7

Store 10

43: store what is in register 25 in data\_ mem address 30

Store 10

44: load from data\_mem address6 to register8

Load 10

45: load from data\_mem address7 to register24

Load 111

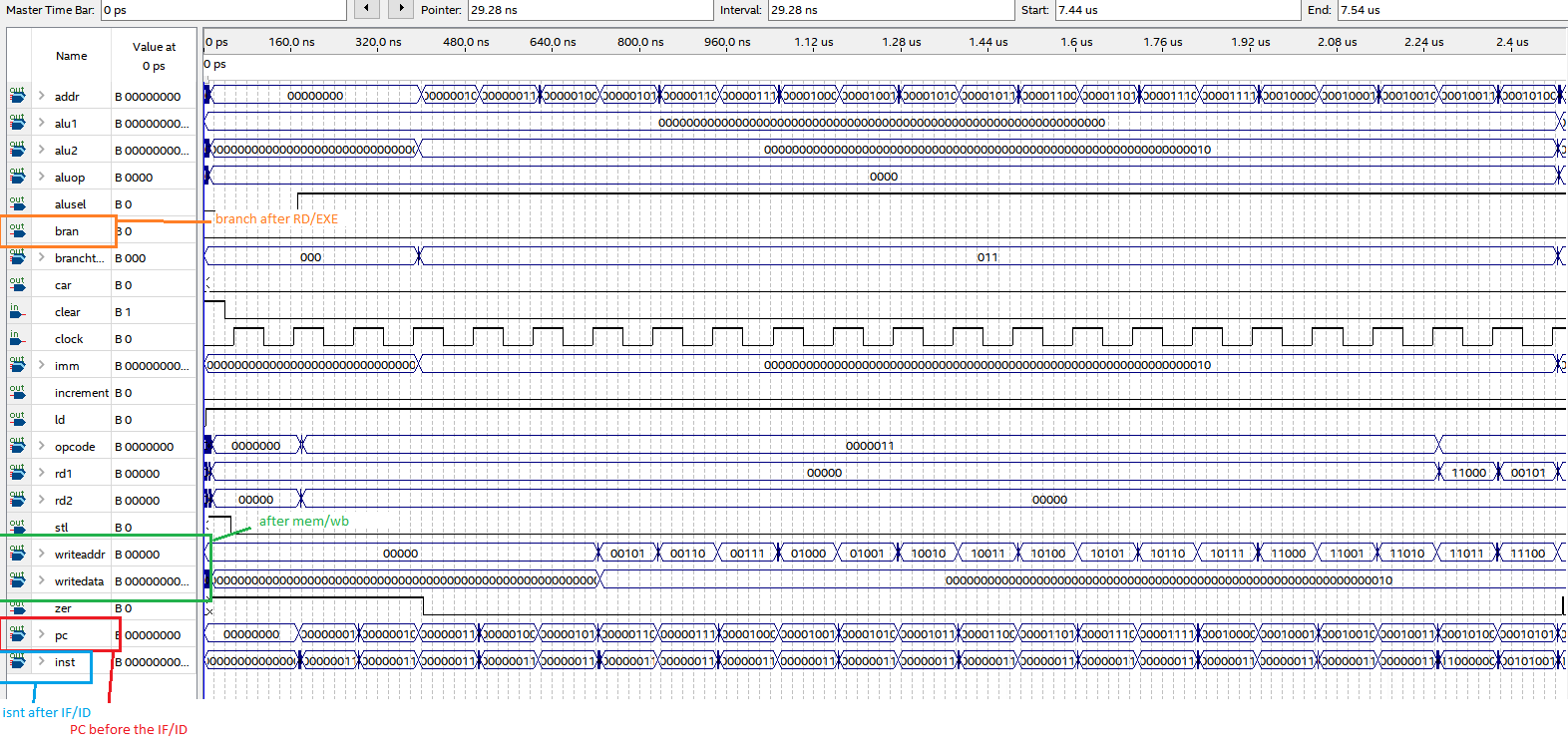
37: load from data\_mem addres30 to register31

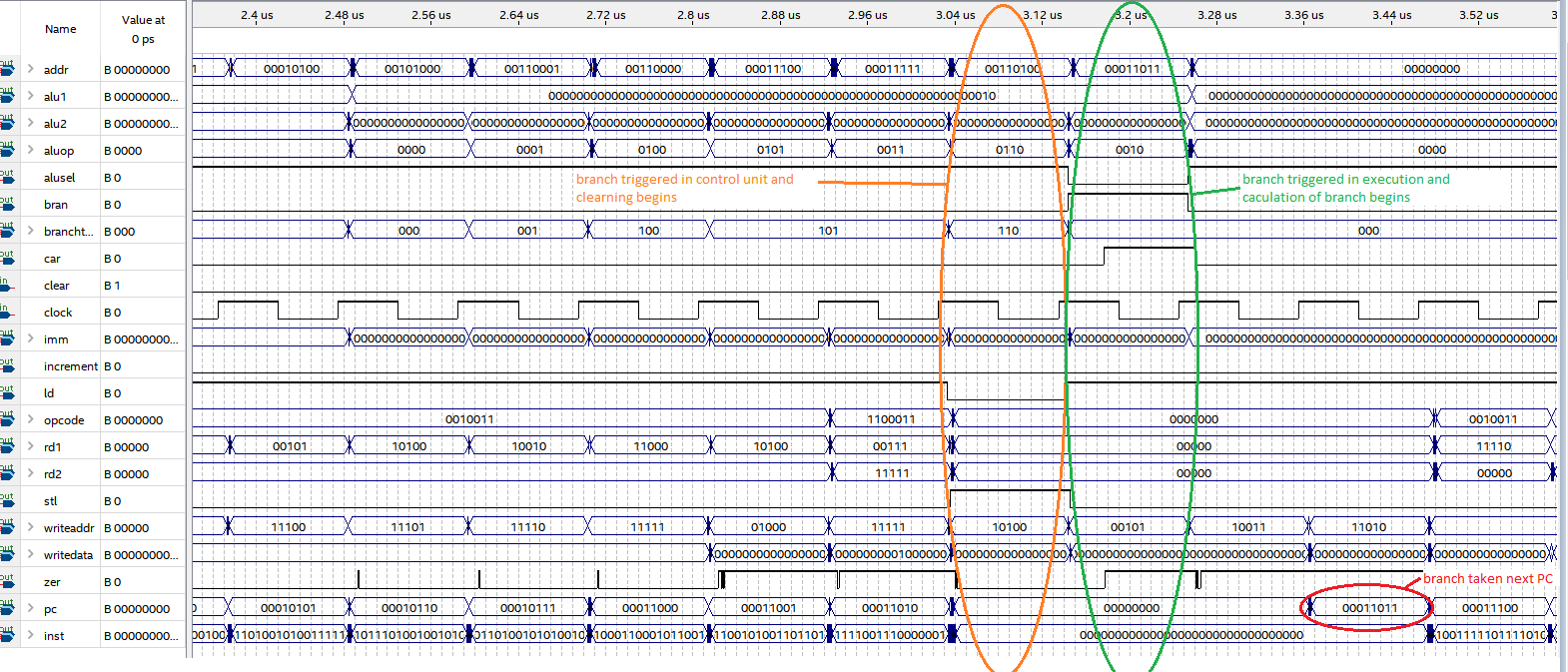
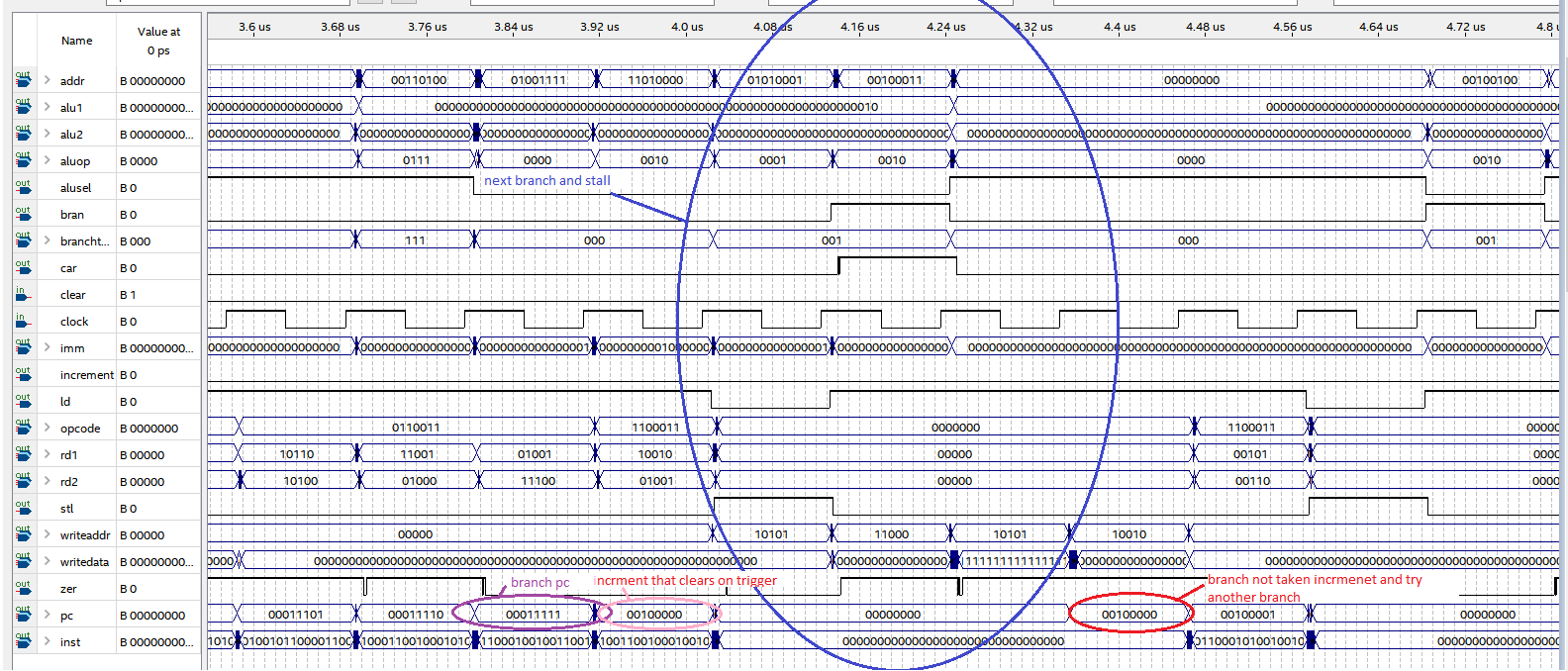
Load 11110

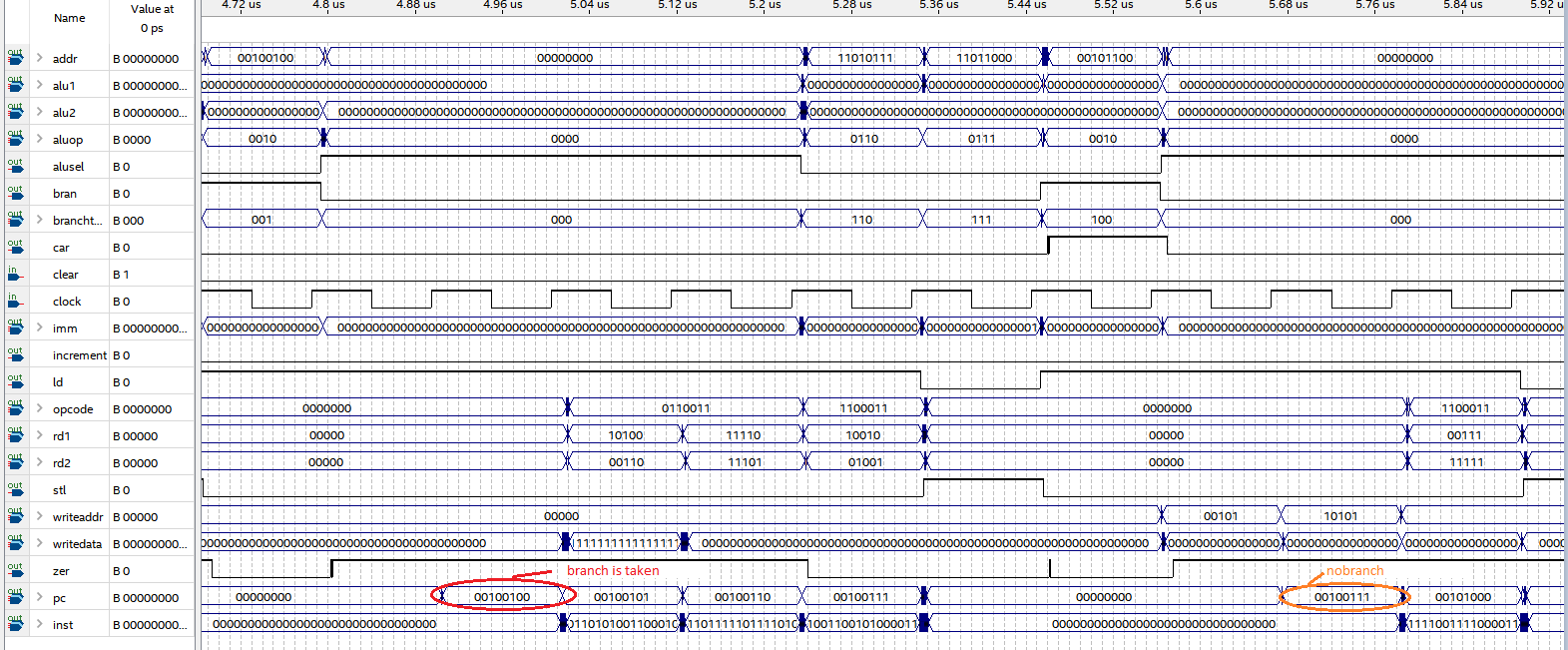
STARTING THE C CODE

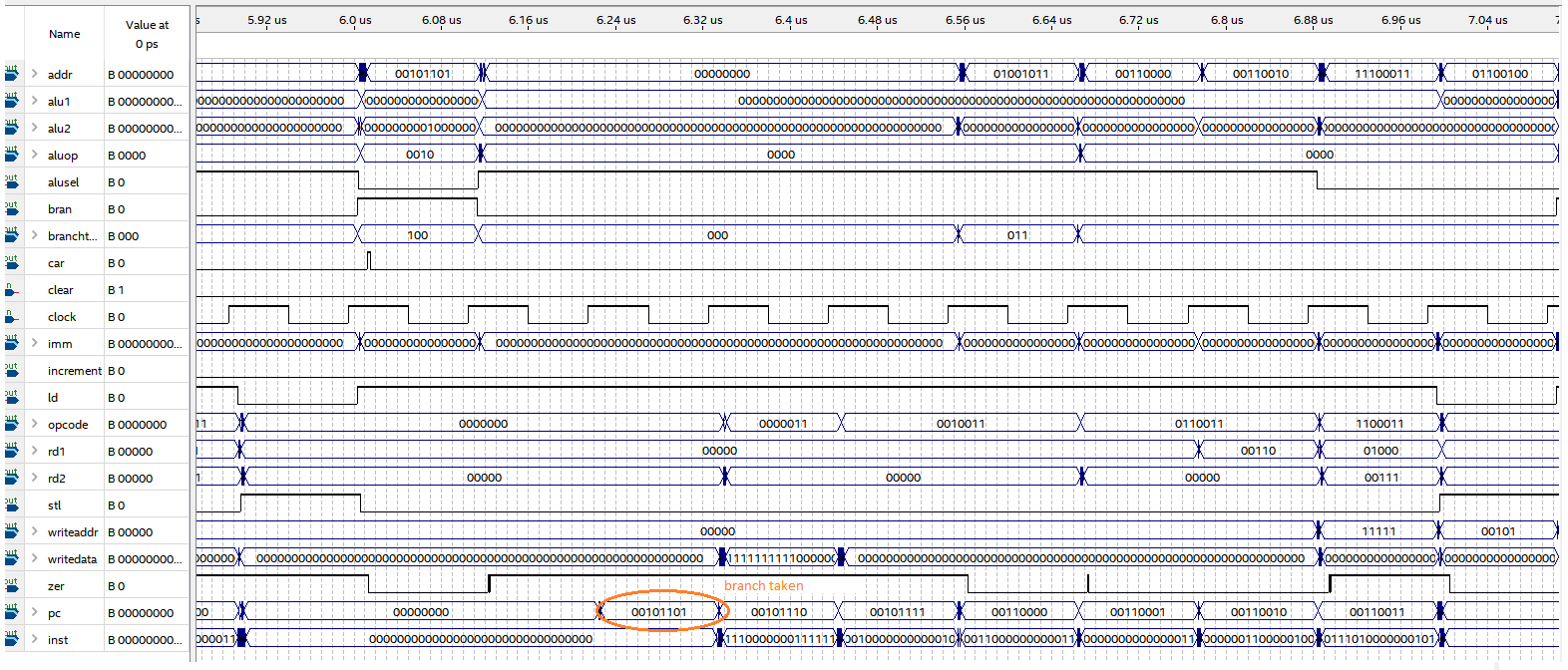
This part is no longer up to date as we need forwarding now to have registers available to read on time

Waveform:







Datapath:

