Lab Assignment 2, Stage 4: Support for all DP opcodes

Kushal Kumar Gupta

2020CS10355

The task at this stage is simply to test the design of stage 3 extensively, covering all DP instructions.

Files, entities and Architectures:

Changes from stage 3-

`Processor_stage4.vhd`: made an error earlier, now DP immediate constant is considered unsigned.

`conditionChecker_stage4.vhd`: Provided support for all 16 predications. Hence, now can use all branch instructions.

How to use:

On edaplayground.com, upload testbench.vhd and run.do in the left column, and ALU_stage3.vhd, RegFile_stage1.vhd, Mem_stage3.vhd, mytypes_stage2.vhd , decoder_stage2.vhd, FlagUpdater_stage3.vhd, conditionChecker_stage3.vhd, programCounter_stage3.vhd, processor_stage3.vhd. Copy contents in design.vhd section as given in the design.vhd file. Select Testbench + Design as VHDL. Then, for-

1.)Simulation

Type testbench in the Top entity. Select Aldec Riviera Pro 2020.04 simulator to simulate the design. Set the run time accordingly and select the EPWave option. Then Save and Run the simulation to get waves of the signals defined in these modules.

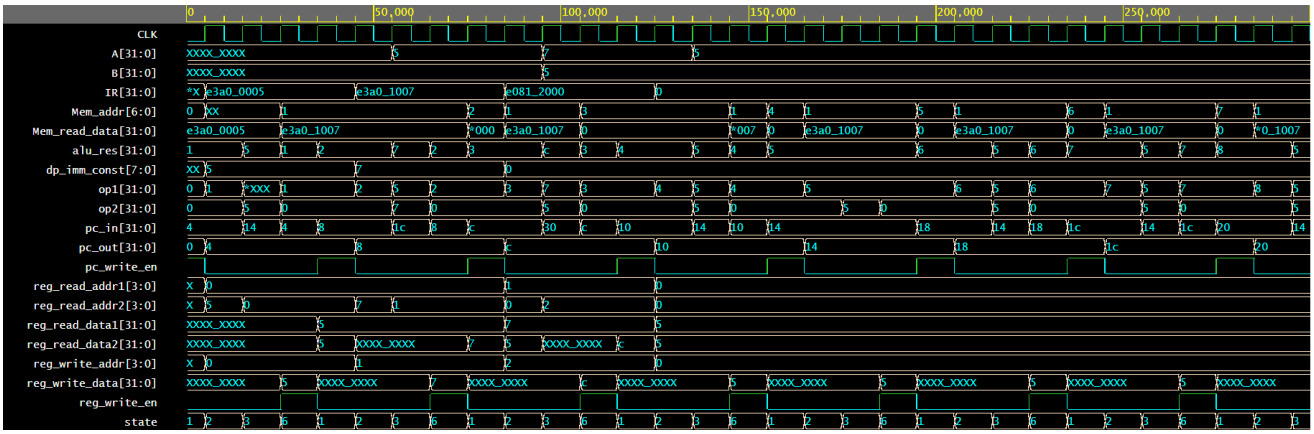
Results of EPWave (Simulation):

Can see the input and output signals of processor against the clock.

Memory initialisation with different programs and EPWave-

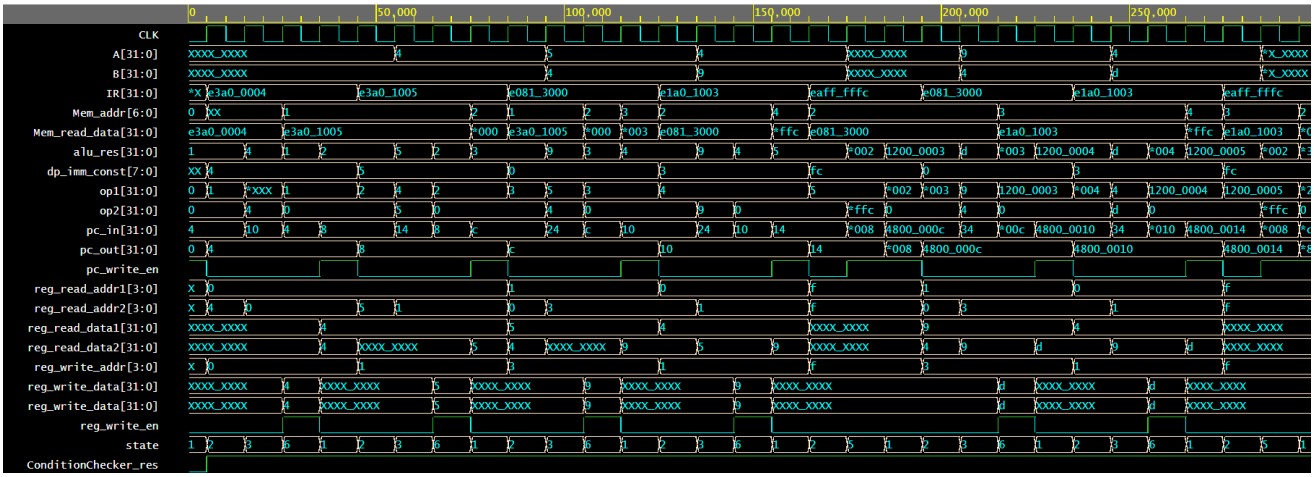
1.

```
signal Mem_space : type_mem := (0 => X"E3A00005",
                                1 => X"E3A01007",
                                2 => X"E0812000",
                                others => X"00000000"
                                );
--mov r0, #5
--mov r1, #7
--add r2, r1, r0
```



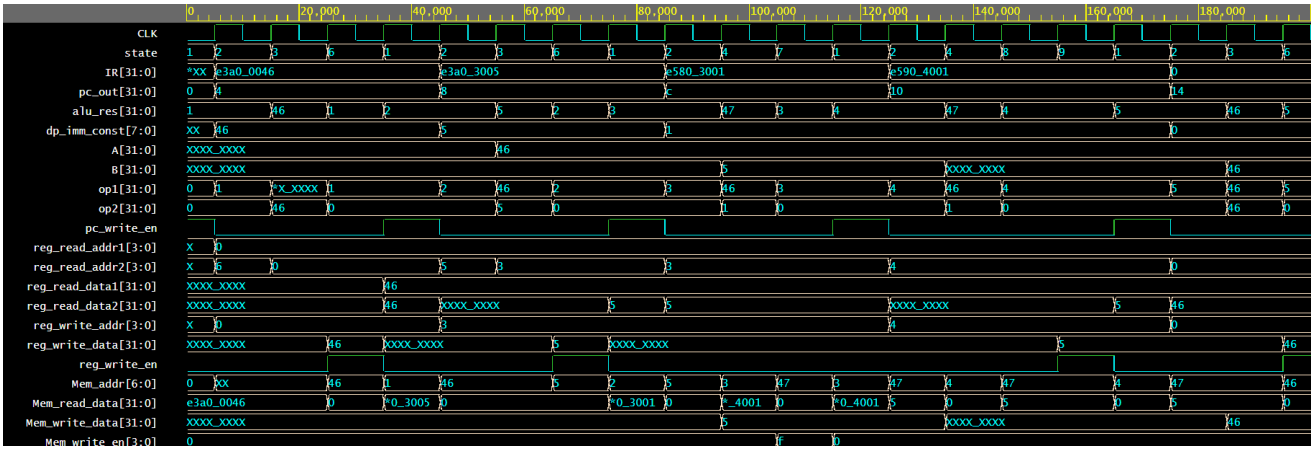
2.

```
signal Mem_space : type_mem := (0 => X"E3A00004",
                                1 => X"E3A01005",
                                2 => X"E0813000",
                                3 => X"E1A01003",
                                4 => X"EAFFFFFC",
                                others => X"00000000"
                                );
--mov r0, #4
--mov r1, #5
--L: add r3, r1, r0
--mov r1, r3
--b L
```



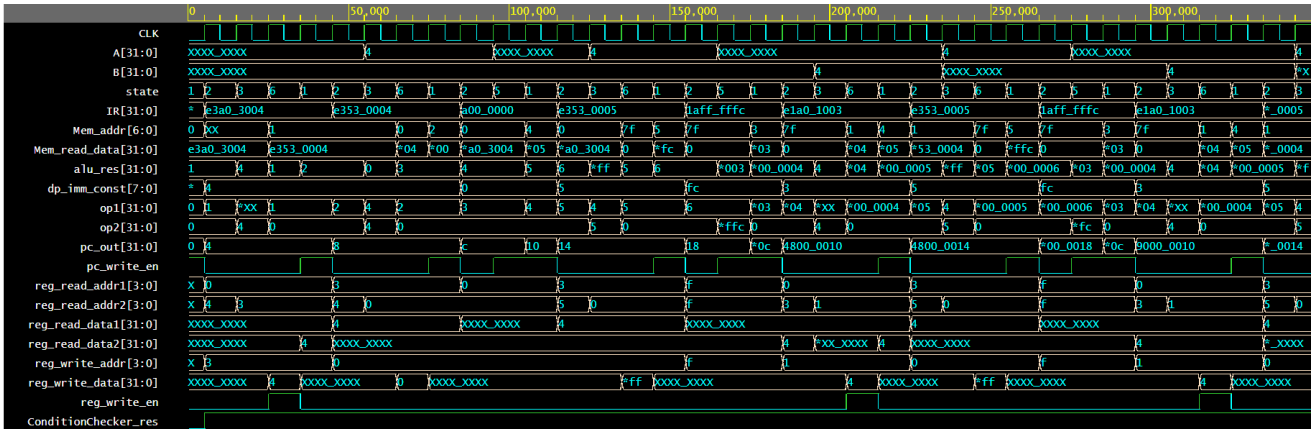
3.

```
signal Mem_space : type_mem := (0 => X"E3A00046",
                                1 => X"E3A03005",
                                2 => X"E5803001",
                                3 => X"E5904001",
                                others => X"00000000"
                                );
--mov r0, #70
--mov r3, #5
--str r3, [r0, #1] --store in word number 71 in memory
--ldr r4, [r0, #1]
```



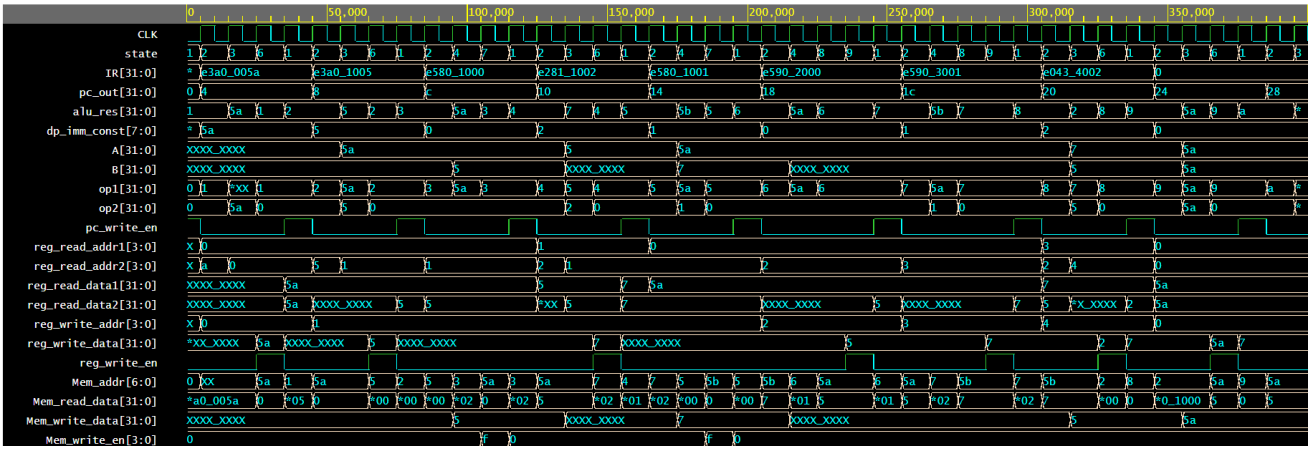
4.

```
signal Mem_space : type_mem := (0 => X"E3A03004",
                                1 => X"E3530004",
                                2 => X"0A000000",
                                3 => X"E1A01003",
                                4 => X"E3530005",
                                5 => X"1AFFFFFC",
                                others => X"00000000"
                                );
--mov r3, #4
--cmp r3, #4
--beq L
--K: mov r1, r3
--L: cmp r3, #5
--bne K
```



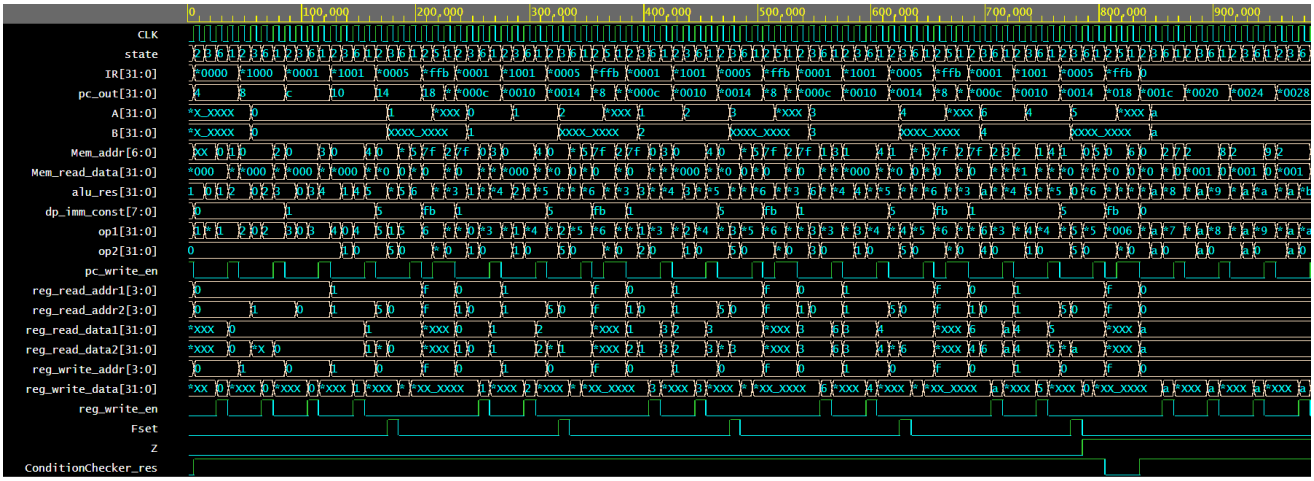
5.

```
signal Mem_space : type_mem :=
                                (0 => X"E3A0005a",
                                 1 => X"E3A01005",
                                 2 => X"E5801000",
                                 3 => X"E2811002",
                                 4 => X"E5801001",
                                 5 => X"E5902000",
                                 6 => X"E5903001",
                                 7 => X"E0434002",
                                 others => X"00000000"
                                 );
--mov r0, #90
--mov r1, #5
--str r1, [r0]
--add r1, r1, #2
--str r1, [r0, #1]
--ldr r2, [r0]
--ldr r3, [r0, #1]
--sub r4, r3, r2
```



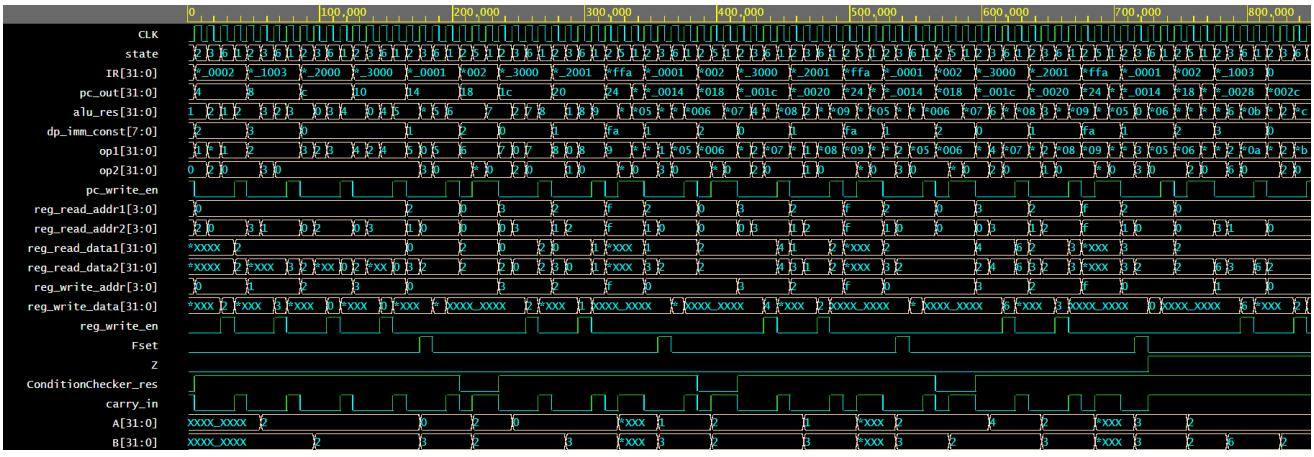
6.

```
signal Mem_space : type_mem :=
                                (0 => X"E3A00000",
                                 1 => X"E3A01000",
                                 2 => X"E0800001",
                                 3 => X"E2811001",
                                 4 => X"E3510005",
                                 5 => X"1AFFFFFB",
                                others => X"00000000"
                                );
--mov r0, #0
--mov r1, #0
--Loop: add r0, r0, r1
--add r1, r1, #1
--cmp r1, #5
--bne Loop
```



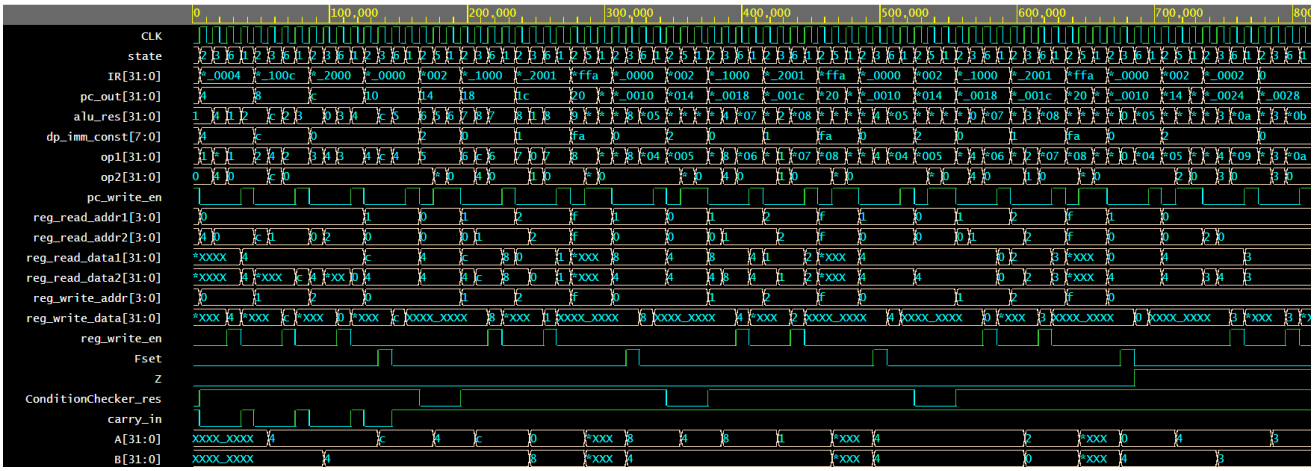
7.

```
signal Mem_space : type_mem :=
                                (0 => X"E3A00002",
                                 1 => X"E3A01003",
                                 2 => X"E3A02000",
                                 3 => X"E3A03000",
                                 4 => X"E1520001",
                                 5 => X"0A000002",
                                 6 => X"E0833000",
                                 7 => X"E2822001",
                                 8 => X"EAFFFFFA",
                                 9 => X"E1A01003",
                                 others => X"00000000");
--Code for multiplying r0*r1
--0  mov r0, #2
--1  mov r1, #3
--2  mov r2, #0
--3  mov r3, #0
--4  B: cmp r2, r1
--5  beq C
--6  add r3, r3, r0
--7  add r2, r2, #1
--8  b B
--9 mov r1, r3 --r1 stores the result
```



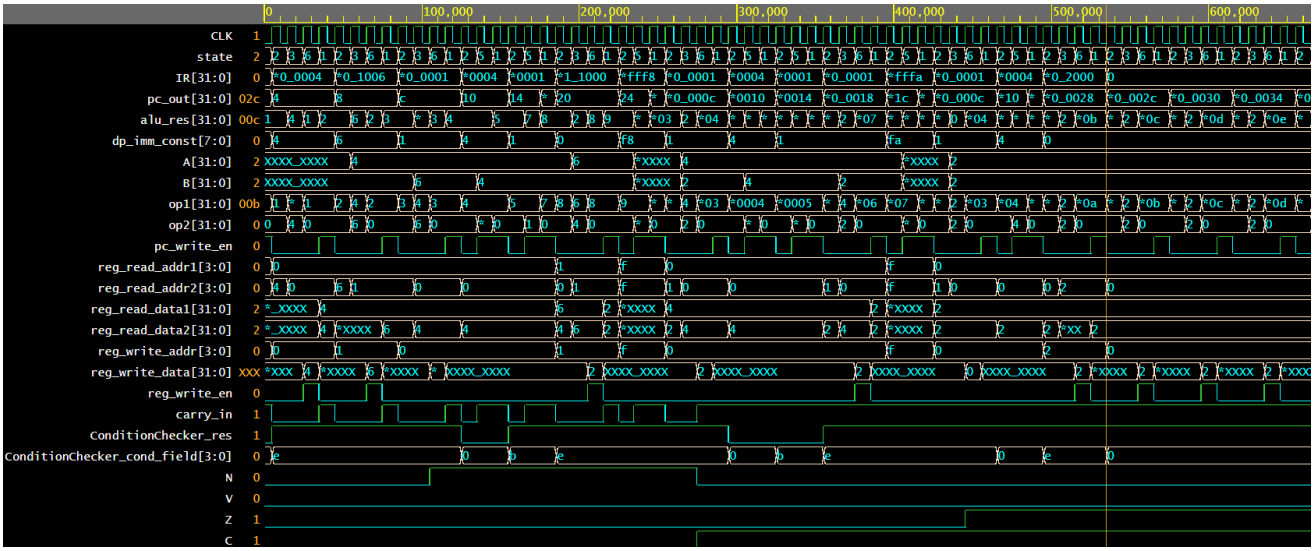
8.

```
signal Mem_space : type_mem :=
                                (0 => X"E3A00004",
                                 1 => X"E3A0100C",
                                 2 => X"E3A02000",
                                 3 => X"E3510000",
                                 4 => X"0A000002",
                                 5 => X"E0411000",
                                 6 => X"E2822001",
                                 7 => X"EAFFFFFA",
                                 8 => X"E1A00002",
                                 others => X"00000000");



--code for dividing r1/r0
--0 mov r0, #4
--1 mov r1, #12
--2 mov r2, #0
--3 a: cmp r1, #0
--4 beq b
--5 sub r1, r1, r0
--6 add r2, r2, #1
--7 b a
--8 b: mov r0, r2 --r0 stores the quotient
```



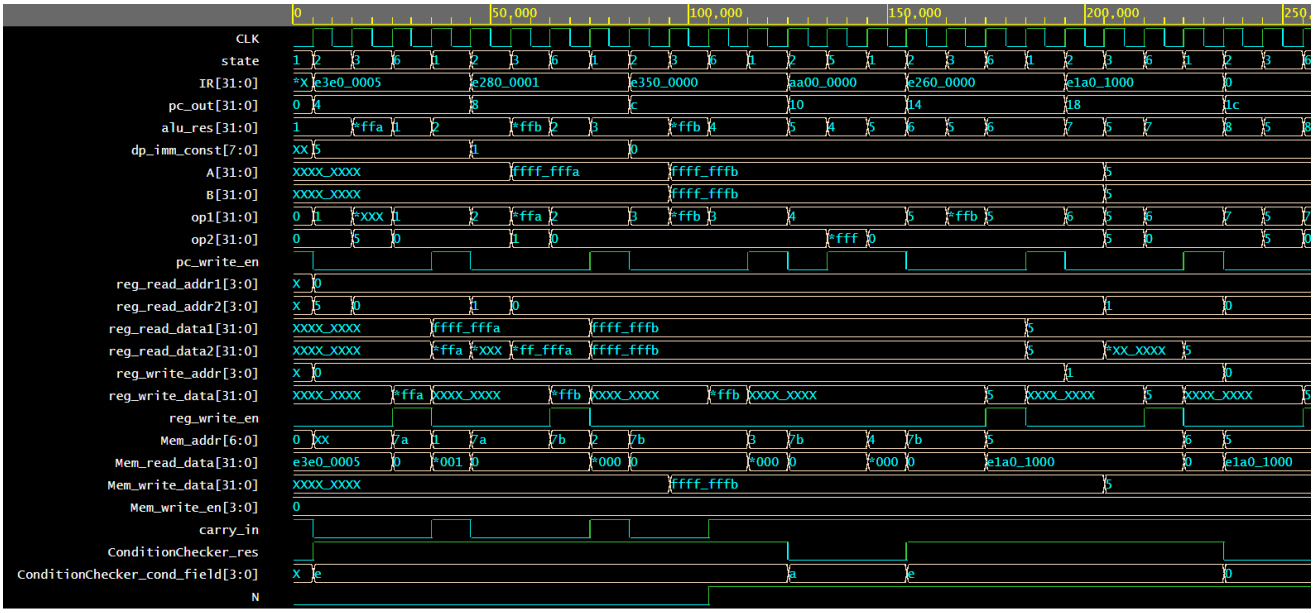
9.

```
signal Mem_space : type_mem :=
                                (0 => X"E3A00004",
                                 1 => X"E3A01006",
                                 2 => X"E1500001",
                                 3 => X"0A000004",
                                 4 => X"BA000001",
                                 5 => X"E0400001",
                                 6 => X"EAFFFFFA",
                                 7 => X"E0411000",
                                 8 => X"EAFFFFF8",
                                 9 => X"E1A02000",
                                others => X"00000000");

--code for finding GCD of r0 and r1 and store result in r2
--0 mov r0, #4
--1 mov r1, #6
--2 A: cmp r0, r1
--3 beq B
--4 blt C
--5 sub r0, r0, r1
--6 b A
--7 C: sub r1, r1, r0
--8 b A
--9 B: mov r2, r0
```



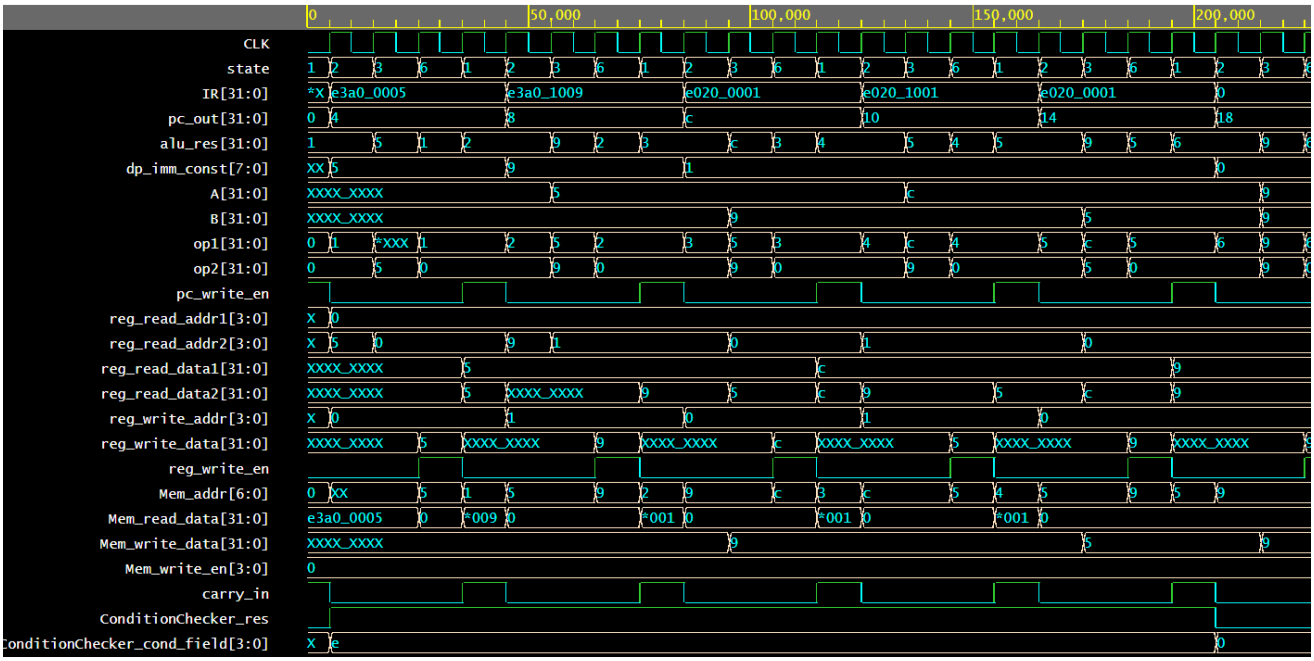
10.

```
    signal Mem_space : type_mem :=
                                    (0 => X"E3E00005",
                                     1 => X"E2800001",
                                     2 => X"E3500000",
                                     3 => X"AA000000",
                                     4 => X"E2600000",
                                     5 => X"E1A01000",
                                     others => X"00000000");

    --code for finding absolute value of r0 (in this example stored -5
in r0 in the first 2 steps)
    --0 mvn r0, #5
    --1 add r0, r0, #1
    --2 cmp r0, #0
    --3 bge N
    --4 rsb r0, r0, #0
    --5 N: mov r1, r0
```



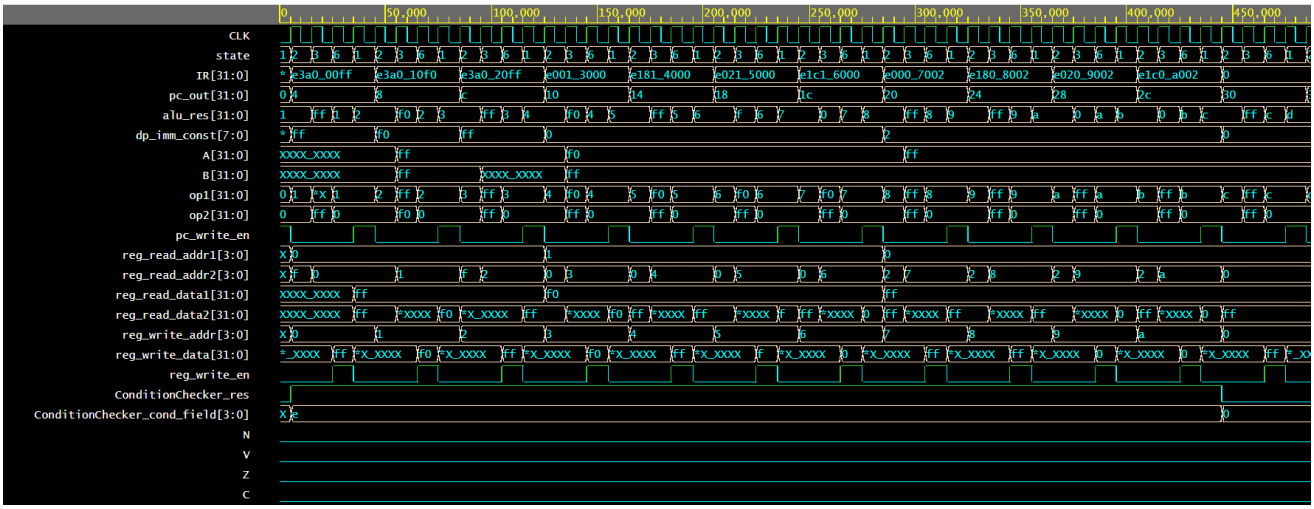
11.

```
signal Mem_space : type_mem :=
                                (0 => X"E3A00005",
                                 1 => X"E3A01009",
                                 2 => X"E0200001",
                                 3 => X"E0201001",
                                 4 => X"E0200001",
                                 others => X"00000000");

--code for swapping registers r0 and r1
-- mov r0, #5
-- mov r1, #9
-- eor r0, r0, r1
-- eor r1, r0, r1
-- eor r0, r0, r1
```



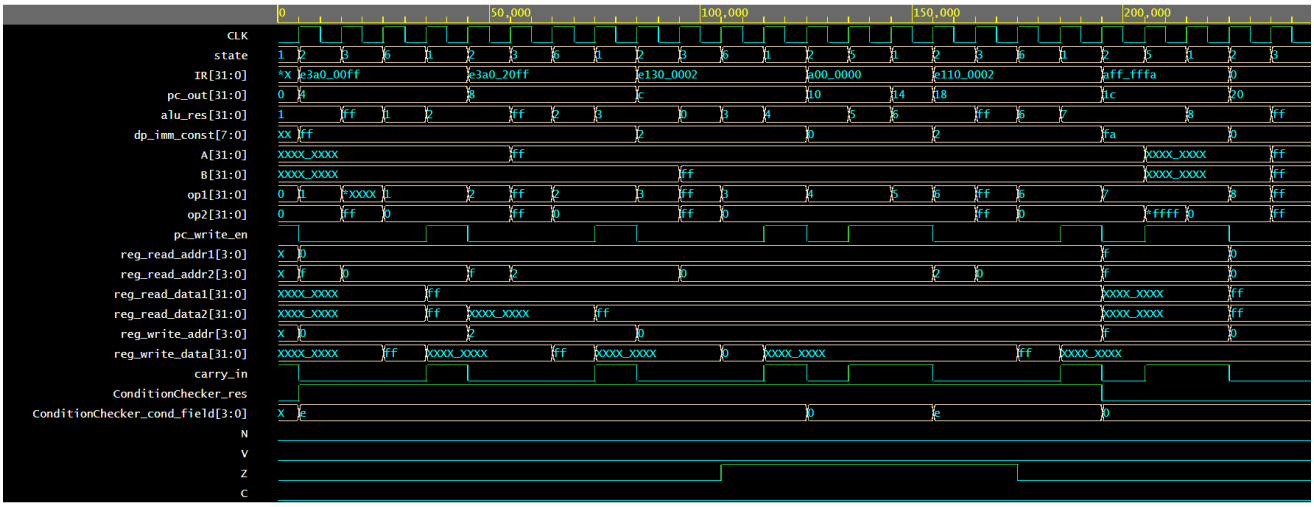
12.

```
signal Mem_space : type_mem :=
                                (0 => X"E3A000FF",
                                 1 => X"E3A010F0",
                                 2 => X"E3A020FF",
                                 3 => X"E0013000",
                                 4 => X"E1814000",
                                 5 => X"E0215000",
                                 6 => X"E1C16000",
                                 7 => X"E0007002",
                                 8 => X"E1808002",
                                 9 => X"E0209002",
                                 10 => X"E1C0A002",
                                 others => X"00000000");

--code for testing logical instructions
-- mov r0, #0xFF
-- mov r1, #0xF0
-- mov r2, #0xFF
-- and r3, r1, r0
-- orr r4, r1, r0
-- eor r5, r1, r0
-- bic r6, r1, r0
-- and r7, r0, r2
-- orr r8, r0, r2
-- eor r9, r0, r2
-- bic r10, r0, r2
```



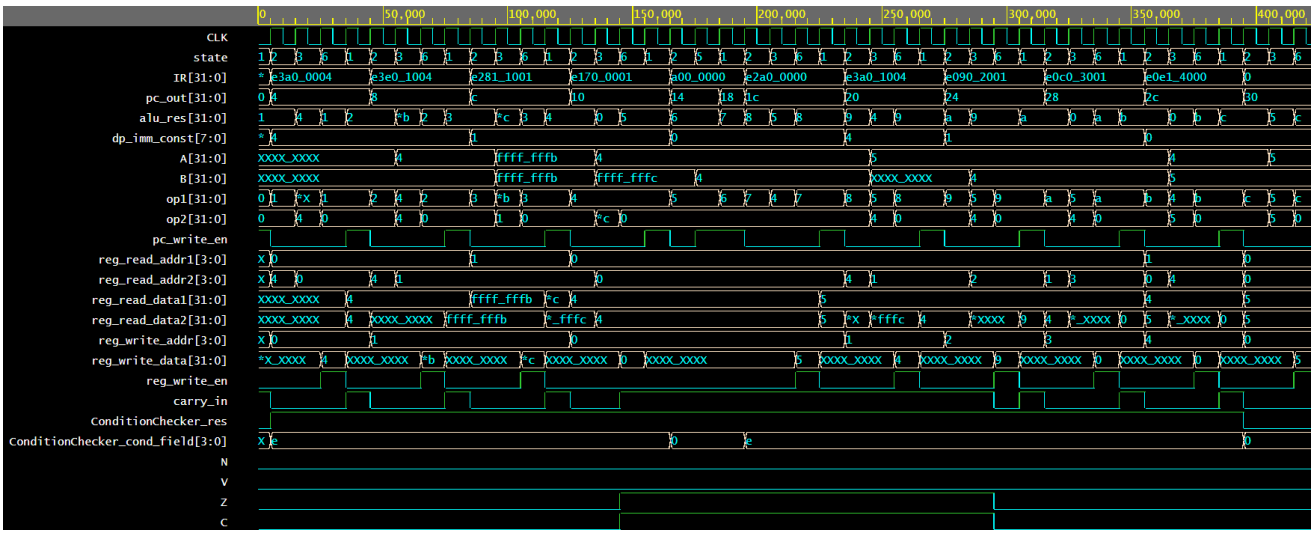
13.

```
signal Mem_space : type_mem :=
                                (0 => X"E3A000FF",
                                 1 => X"E3A020FF",
                                 2 => X"E1300002",
                                 3 => X"0A000000",
                                 4 => X"E1A01000",
                                 5 => X"E1100002",
                                 6 => X"0AFFFFFA",
                                 others => X"00000000");

--code for testing test instructions
-- mov r0, #0xFF
-- mov r2, #0xFF
-- N: teq r0, r2
-- beq K
-- mov r1, r0
-- K: tst r0, r2
-- beq N
```



14.

```
signal Mem_space : type_mem :=
                                (0 => X"E3A00004",
                                 1 => X"E3E01004",
                                 2 => X"E2811001",
                                 3 => X"E1700001",
                                 4 => X"0A000000",
                                 5 => X"E0013000",
                                 6 => X"E2A00000",
                                 7 => X"E3A01004",
                                 8 => X"E0902001",
                                 9 => X"E0C03001",
                                 10 => X"E0E14000",
                                 others => X"00000000");

--code for testing adc, rsc, sbc, cmn
--      mov r0, #4
--      mvn r1, #4
--      add r1, r1, #1
--      cmn r0, r1
--      beq A
--      and r3, r1, r0
-- A: adc r0, r0, #0
--      mov r1, #4
--      adds r2, r0, r1
--      sbc r3, r0, r1
--      rsc r4, r1, r0
```



15.

```
signal Mem_space : type_mem :=
                                (0 => X"E3A00003",
                                 1 => X"E3A01009",
                                 2 => X"E0602001",
                                 3 => X"E1500001",
                                 4 => X"E0E02001",
                                 5 => X"E0C12000",
                                 others => X"00000000");

--code for testing rsc, sbc, rsb
-- mov r0, #3
-- mov r1, #9
-- rsb r2, r0, r1
-- cmp r0, r1
-- rsc r2, r0, r1
-- sbc r2, r1, r0
```