CS341 Computer Architecture Lab03

170050053-Sudhir Kumar

Q1. 5 Stage, without forwarding or hazard detection

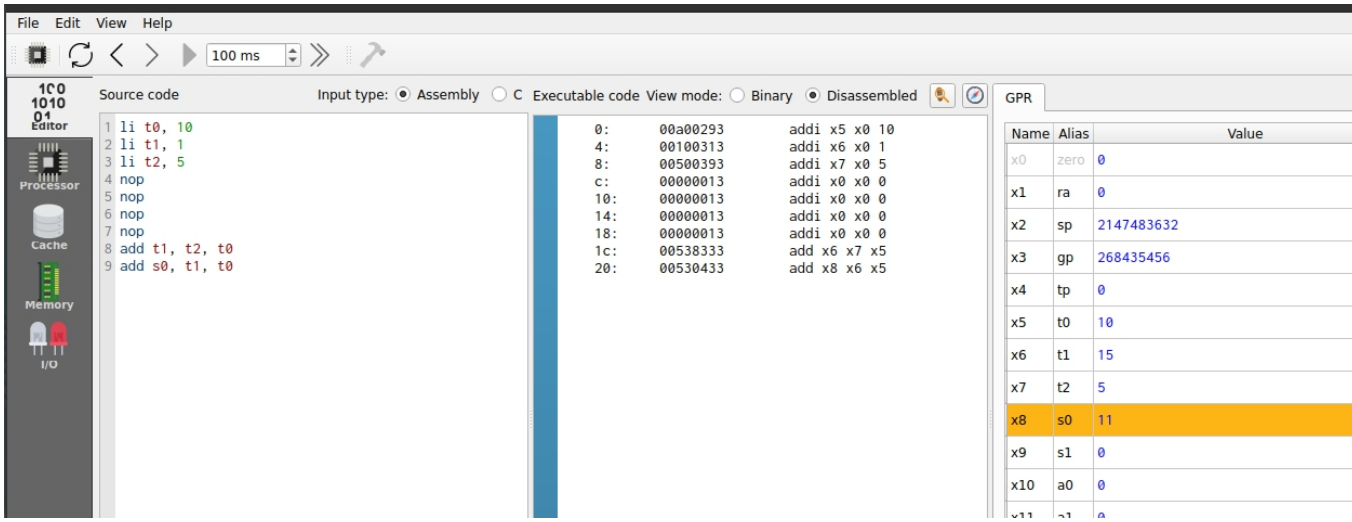

fig:-- without forwarding or hazard detection,
gives wrong value stored in register s0
(Note: line 4,5,6,7 nops are just for security purpose)

because of Read After Write
here instruction line 9 is reading value of register t1, executing at 10th cycle, before getting updated by instruction line 8, writing back at 11th cycle

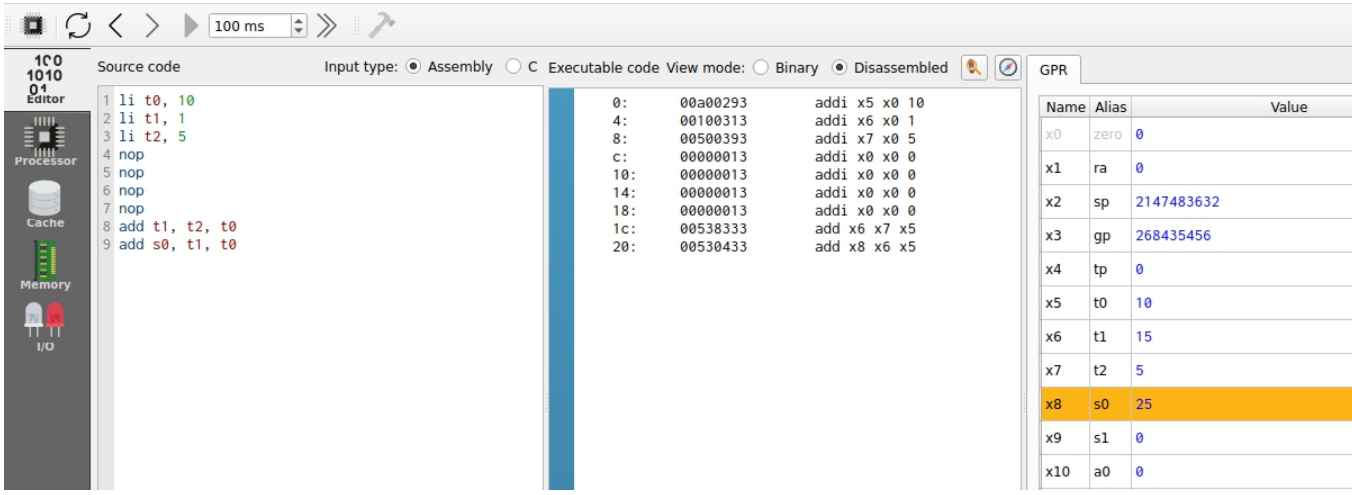

fig:------- 5-stage in-order processor with hazard detection/elimination and forwarding, gives expected value.

the correct final value stored in register s0 should be 25

Q-2. 5 Stage, w/o forwarding, with hazard detection

[a]

instructions:
li t0, 10
li t1, 1
li t2, 5
nop
nop
nop
nop
add t1, t2, t0
lw s0, 0(t1)

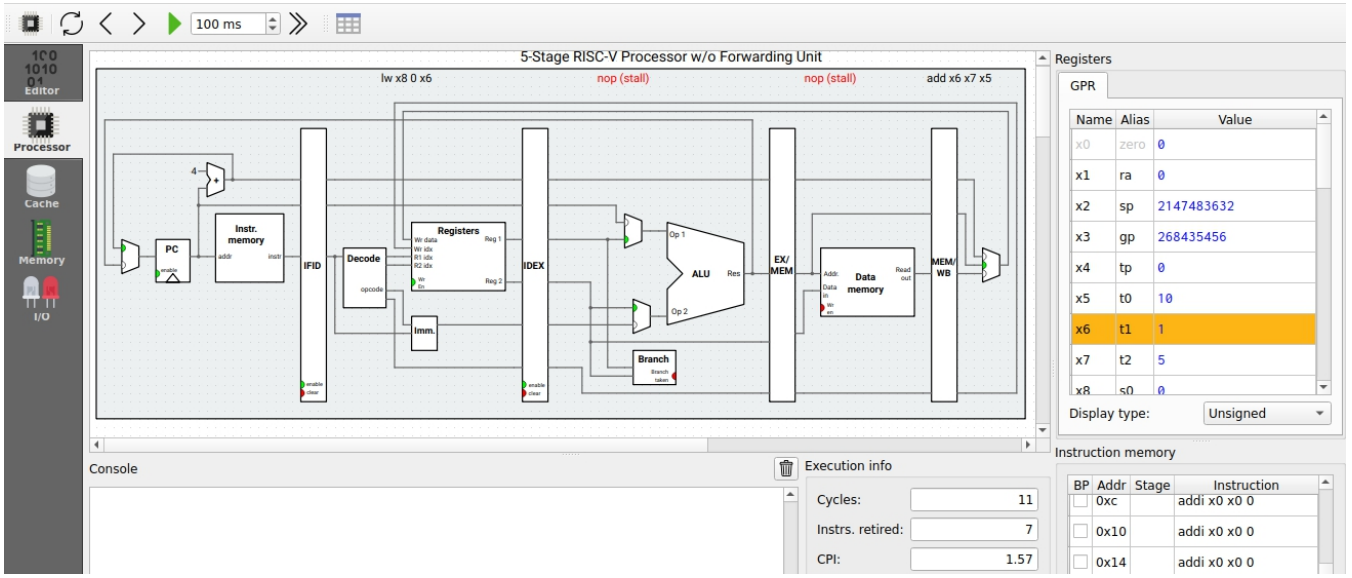

fig:---- nop (stall) , when forwarding unit is not present

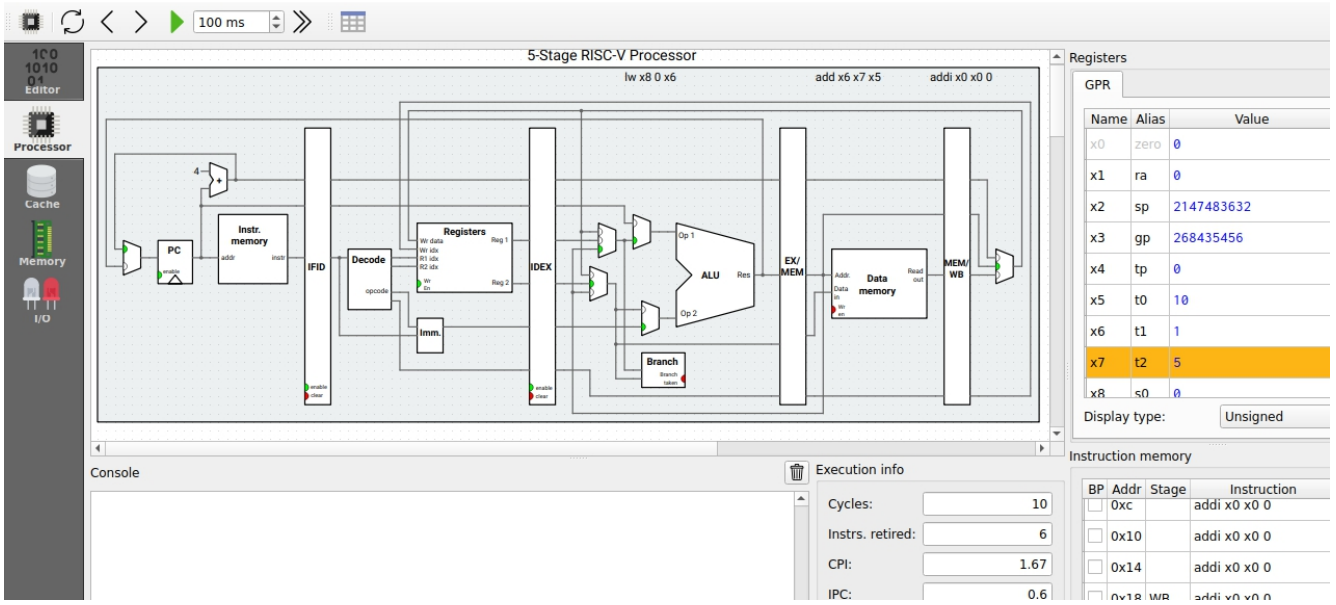

fig:--- no stalls when forwarding (and hazard detection) is enabled
for the same code

[b]

```
li t2, 5
li t0, 10
li t1, 1
nop
nop
nop
nop
add t1, t2, t0

add s0, t1, t0

add s1, t2, t0
add a0, t2, t0
```

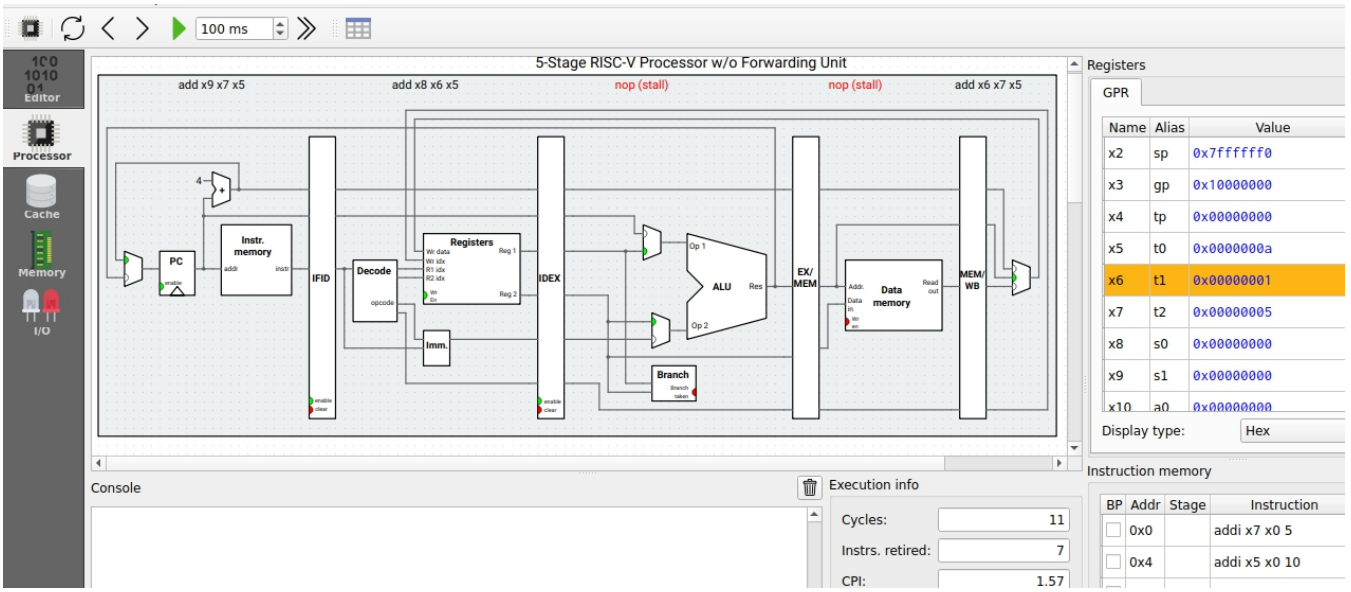

fig:------- initial code(above) needs stalls as screenshot shows

after rearranging of instructions (last two lines is inserted between lines just above it)

```
li t2, 5
li t0, 10
li t1, 1
nop
nop
nop
nop
add t1, t2, t0

add s1, t2, t0
add a0, t2, t0

add s0, t1, t0
```

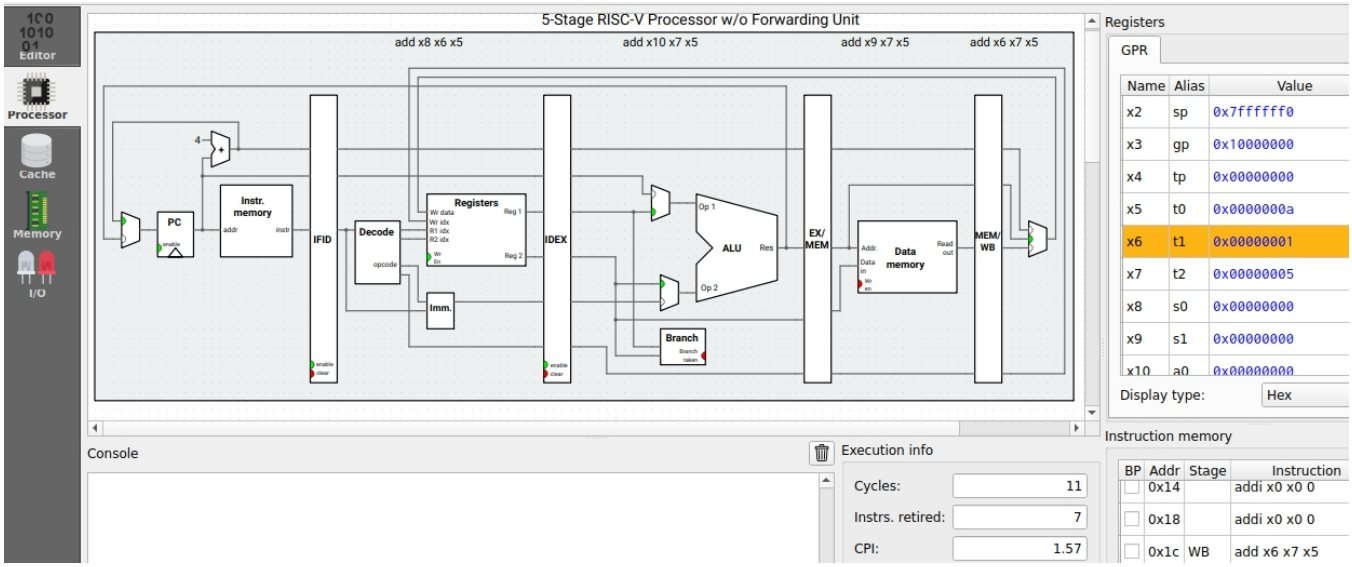

fig:----- after re-arrange of instructions stalls are avoided

here RAW data hazards was likely to happen in first case which was taken care by stalls but after re-arrangement of instructions RAW was removed.

Q-3. Stalls and Forwarding

[a]

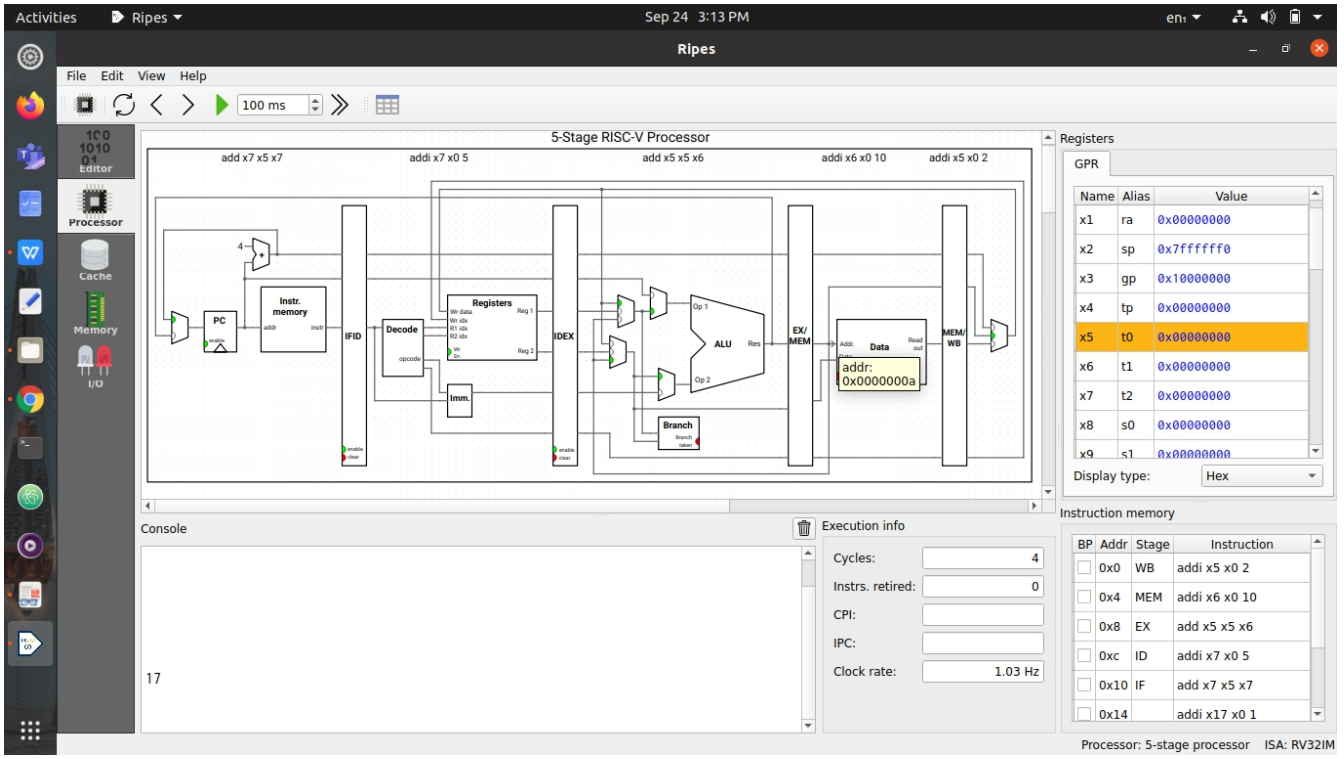

fig:----Address input of data memory = addr: 0x0000000a

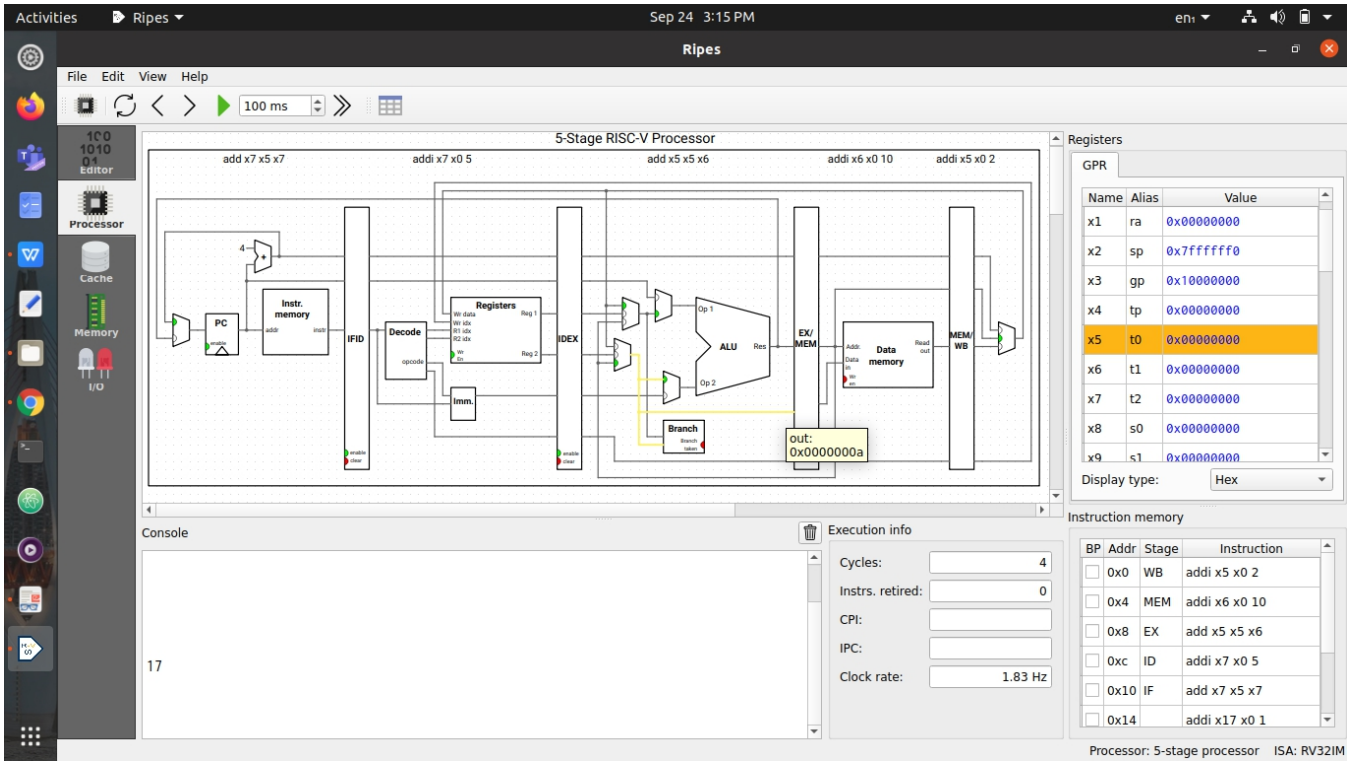

fig:------Address input of EX/MEM pipeline= out: 0x0000000a

[b]

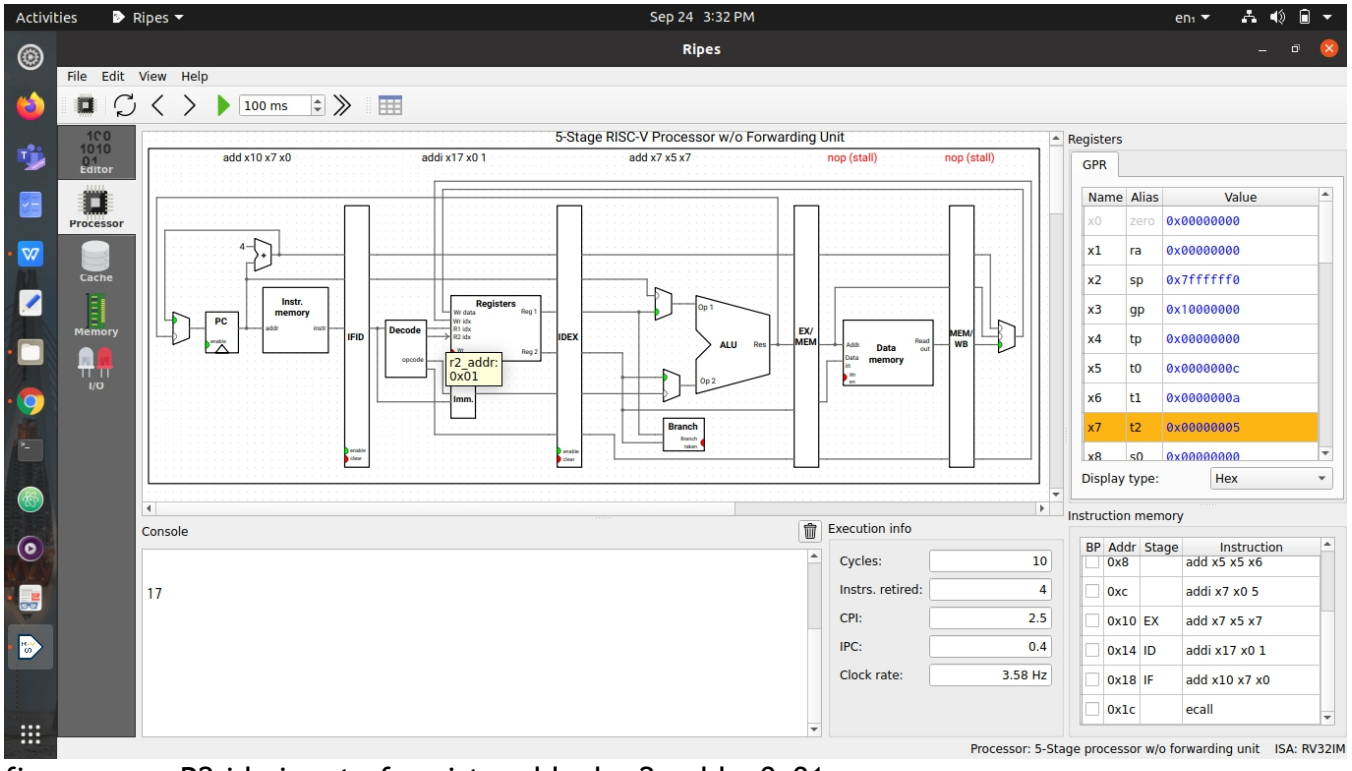

fig: ----------R2 idx input of registers block=r2_addr: 0x01

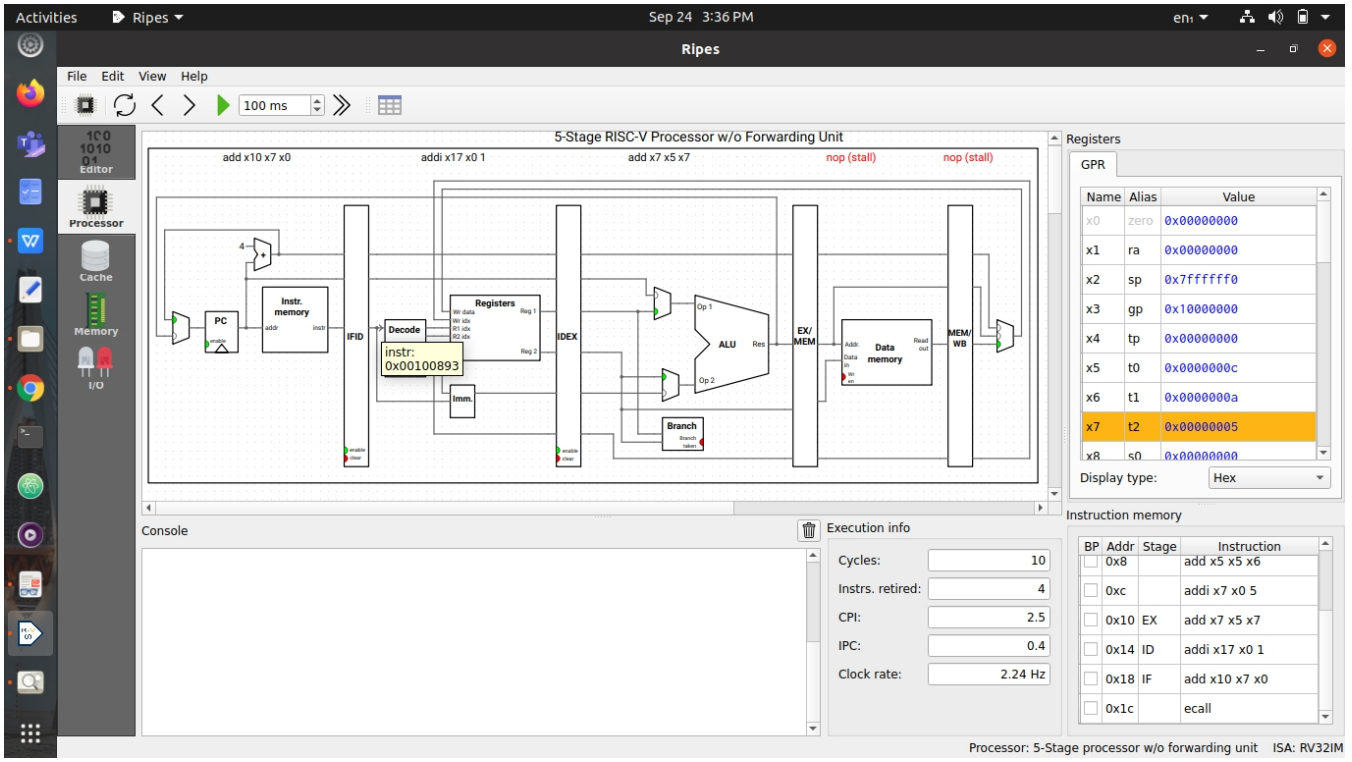

fig:-------- input of decoder= instr: 0x00100893

[c]

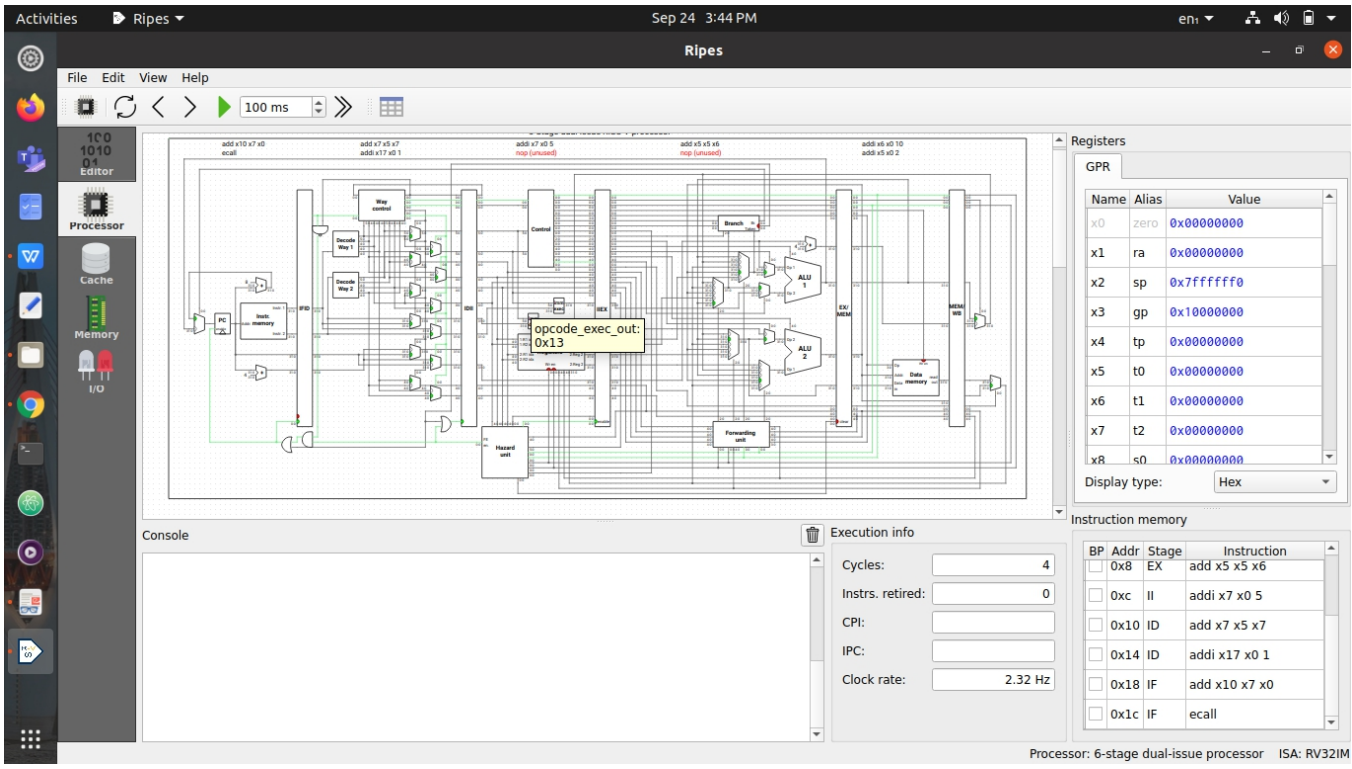

fig:------- value stored in= opcode_exec_out: 0x13

[d]

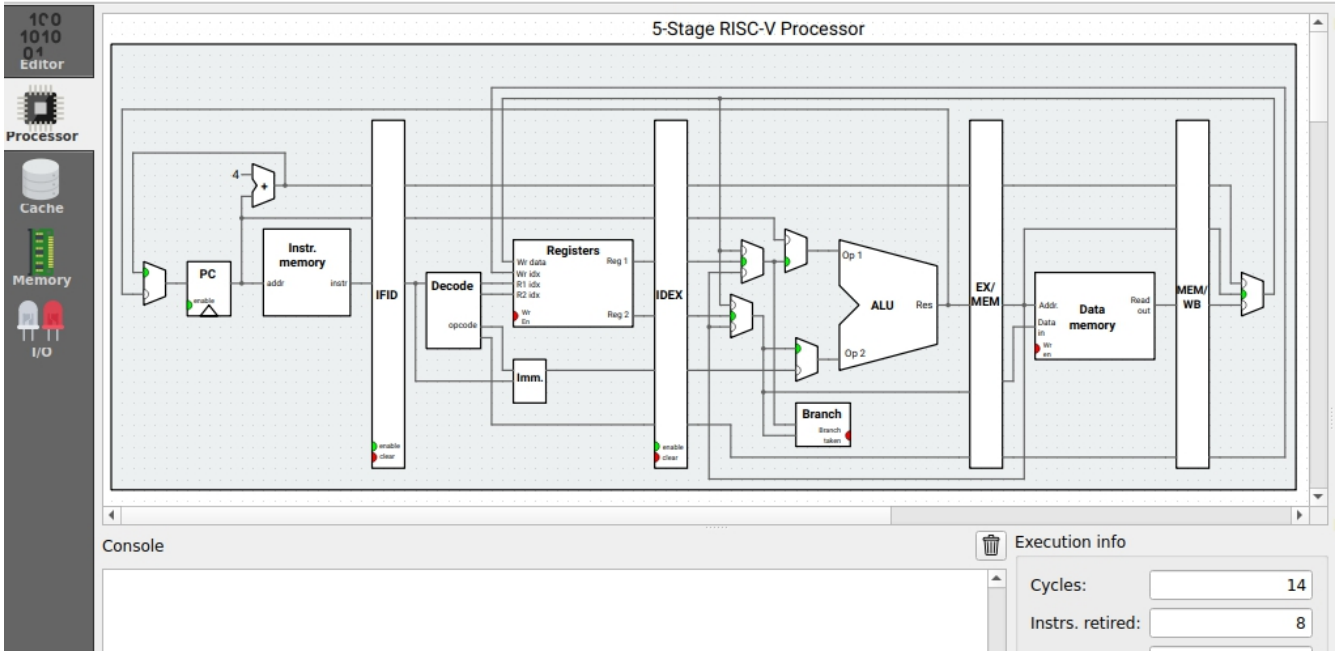

fig:--- 5-stage RISC-V processor detects hazards and has forwarding.
number of cycles needed = 14

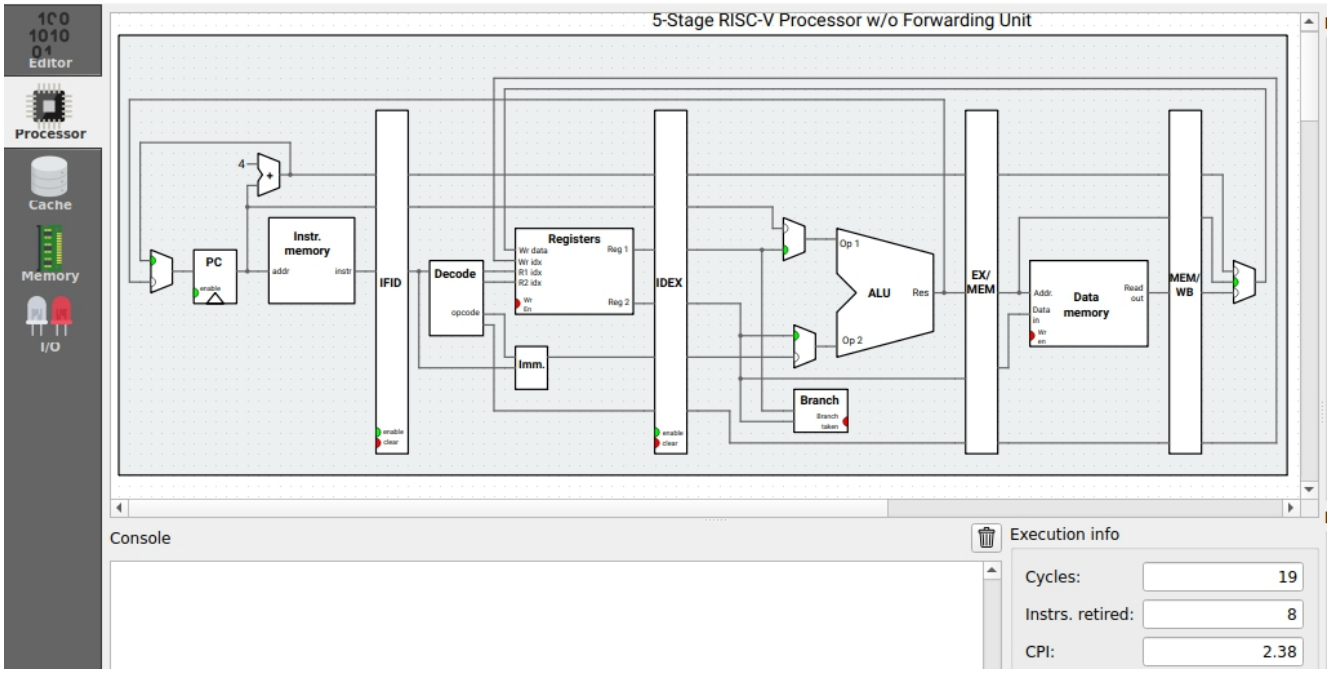

fig:---- 5-stage RISC-V processor, detects hazards and has no forwarding.
number of cycles needed = 19

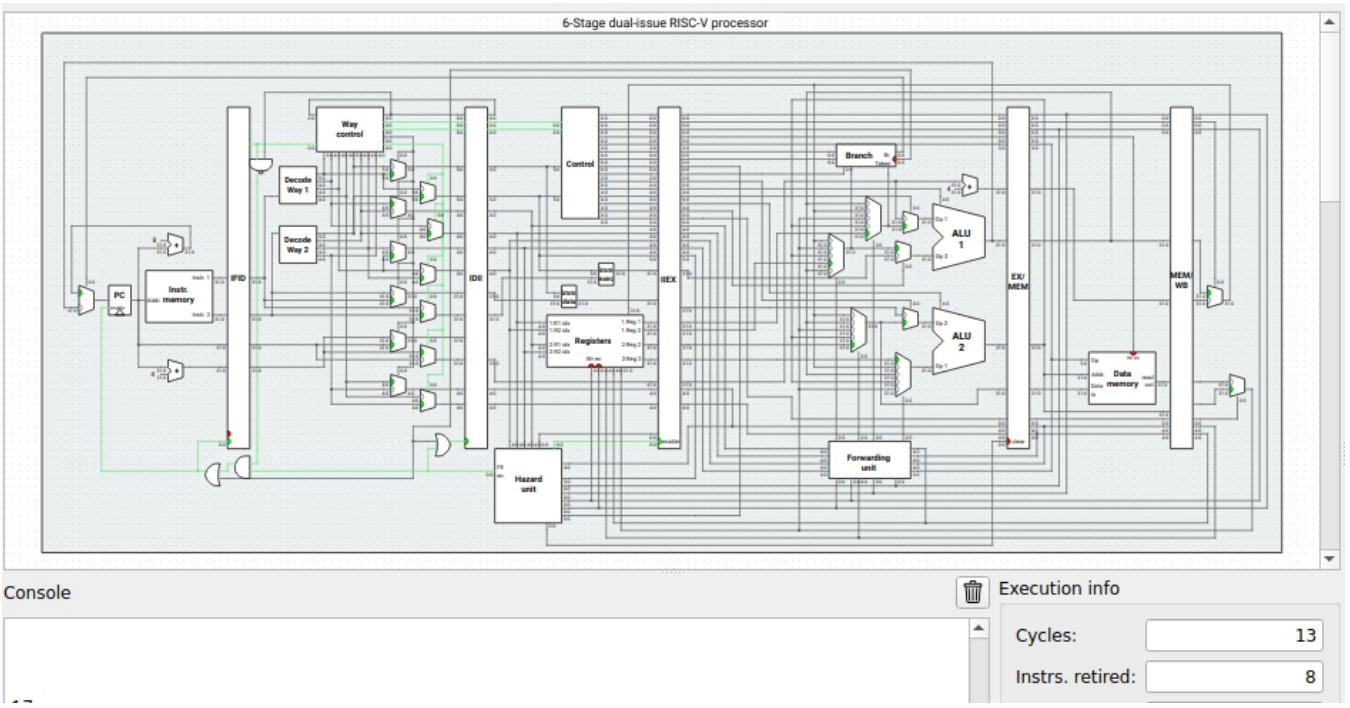

fig:---- 6-stage dual-issue processor
number of cycles needed = 13

[e]

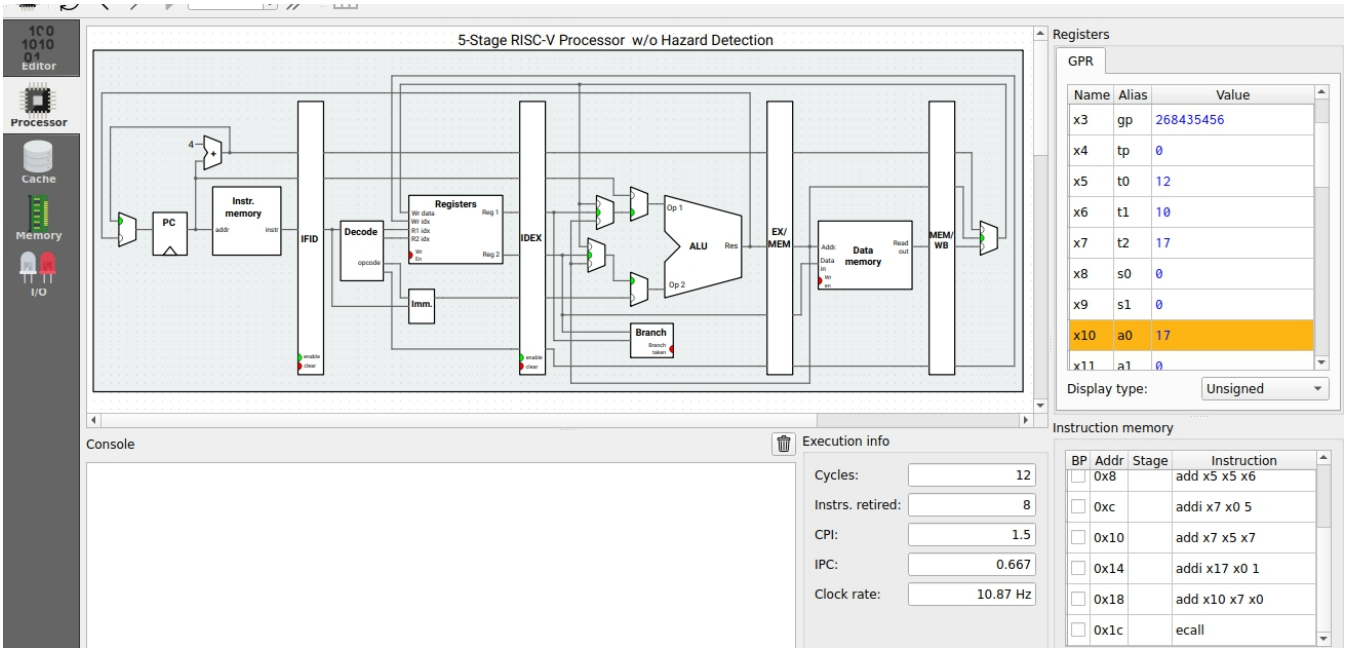

fig:--- **empty output** with an error of system call (value of a7: 0 not good practice)
this is because before we **Write Back** value 1 to a7
in third last instruction(li a7 1),
ecall is executed, assuming a7 is 0
(ecall is not getting updated value of a7)
and hence there is no output

after modification in given instructions

li t0 2
li t1 10
add t0 t0 t1
li t2 5
add t2 t0 t2
li a7 1
add a0 t2 x0
nop
nop
nop
ecall

three nops/stalls are inserted just before ecall
to get updated values of a7 and a0 during ecall

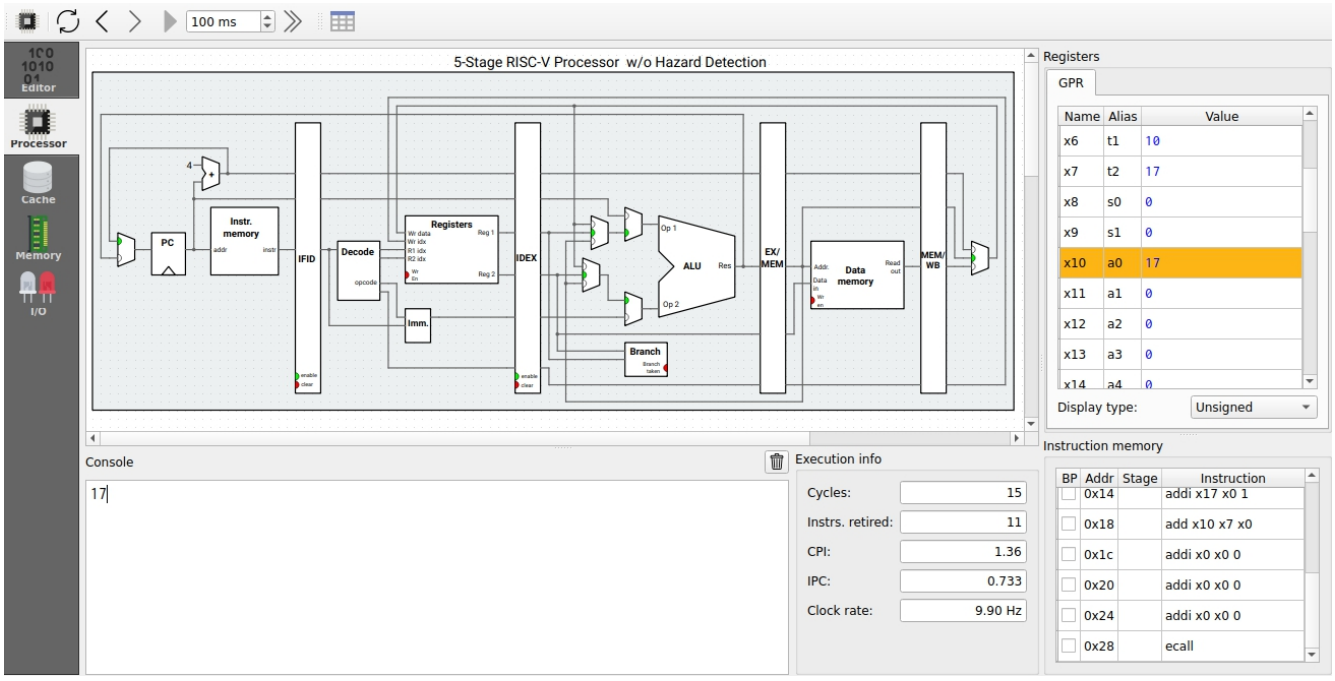

fig:----- after modification output = 17 (as expected)

Q-4. Maximising Efficiency

li t1 1
li t2 2
li a0 3
li s0 4
add t1 t1 t2
li t2 5

```
add a0 a0 s0
li s0 6
add t1 t1 t2
li t2 7
add a0 a0 s0
li s0 8
add t1 t1 t2
li t2 9
add a0 a0 s0
li s0 10
add t1 t1 t2
li a7 1
add a0 a0 s0
add a0 a0 t1
ecall
```

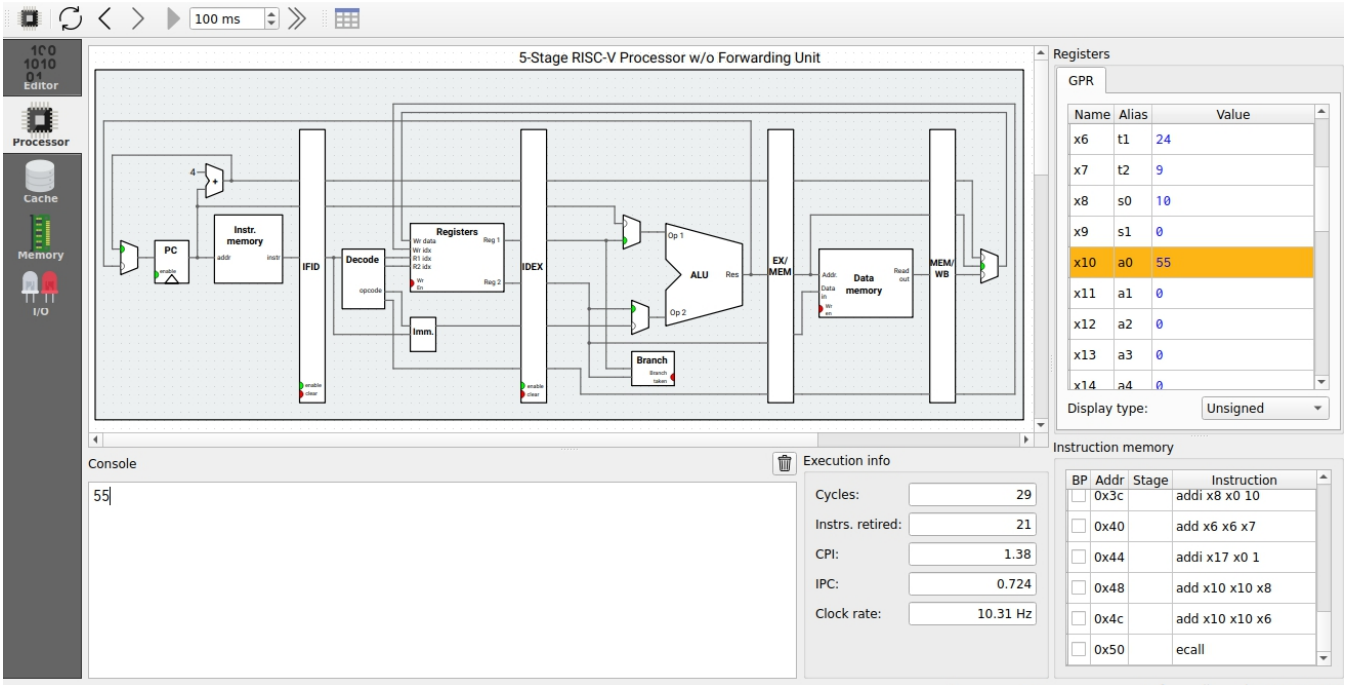

fig:---- risc-v processor, w/o forwarding with hazard detection

number of cycles = 29
number of registers = 5 (t1, t2, a0, s0, a7)