CS232-Lab 4

210050115 - Patil Vipul Sudhir

March 7, 2023

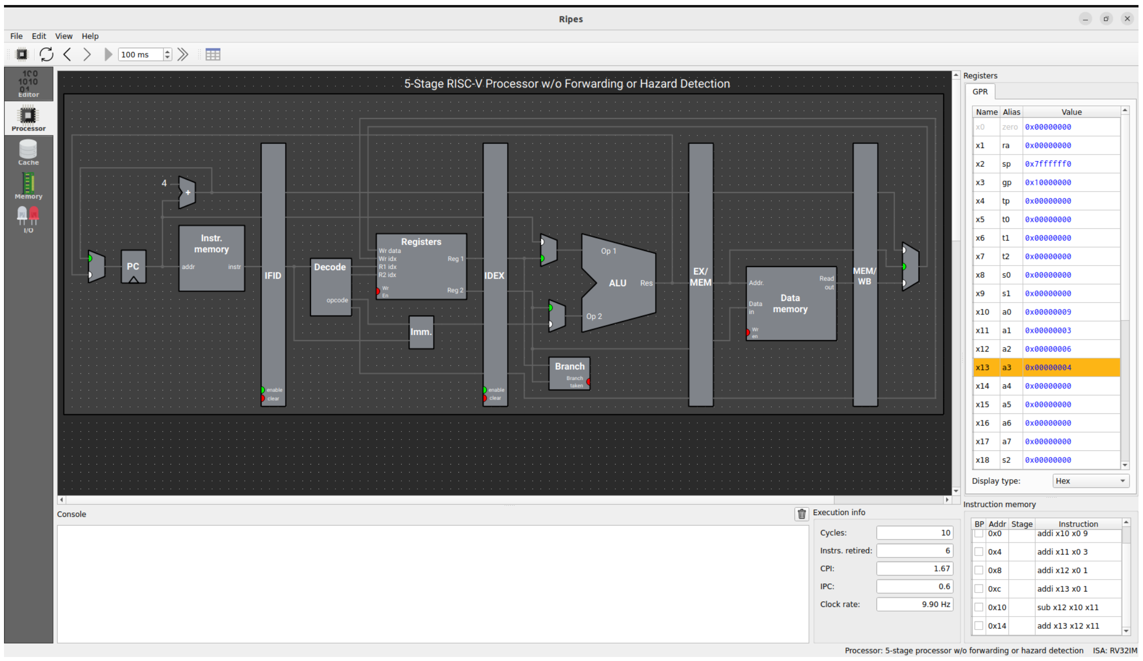
1 Q1

1.1 a. Read after write hazard

- Final answer expected: 9
- Final answer obtained: 4
- Explanation: Firstly initialized 4 registers a0, a1, a2, a3. Then subtracted a1 from a0 and stored it in a2(writing into a2). In next instruction value from a2 is read(reading from a2) and is added with a1 and stored into a3. Before writing up the subtracted value in memory, value from memory is read so previous (old value) got added into a1 to give a3.

```
.text
main:
    li a0, 9        # Initializing a0 = 9
    li a1, 3        # Initializing a1 = 3
    li a2, 1        # Initializing a2 = 1
    li a3, 1        # Initializing a3 = 1
    sub a2, a0, a1  # subtracting a1 from a0
    # value in a2 becomes 6(=9-3) ideally
    add a3, a2, a1  # adding a2 and a1 and storing it into a3
```

Listing 1: Code for q1-a



Registers

GPR

Name	Alias	Value
x0	zero	0x00000000
x1	ra	0x00000000
x2	sp	0x7ffffff0
x3	gp	0x10000000
x4	tp	0x00000000
x5	t0	0x00000000
x6	t1	0x00000000
x7	t2	0x00000000
x8	s0	0x00000000
x9	s1	0x00000000
x10	a0	0x00000009
x11	a1	0x00000003
x12	a2	0x00000006
x13	a3	0x00000004
x14	a4	0x00000000
x15	a5	0x00000000
x16	a6	0x00000000
x17	a7	0x00000000
x18	s2	0x00000000

Display type: Hex

Execution info

Cycles:	10
Instrs. retired:	6
CPI:	1.67
IPC:	0.6
Clock rate:	9.90 Hz

Instruction memory

BP	Addr	Stage	Instruction
	0x0		addi x10 x0 9
	0x4		addi x11 x0 3
	0x8		addi x12 x0 1
	0xc		addi x13 x0 1
	0x10		sub x12 x10 x11
	0x14		add x13 x12 x11

Processor: 5-stage processor w/o forwarding or hazard detection ISA: RV32IM

Figure 1: Q1-a: Read after write hazard

1.2 b. Write after read hazard

- This type of hazard is not possible
- Explanation: It is not possible because the write happens in the last stage of the processor, reading happens in the first stage of the first instruction so writing later in other stage doesn't affect reading in previous instructions

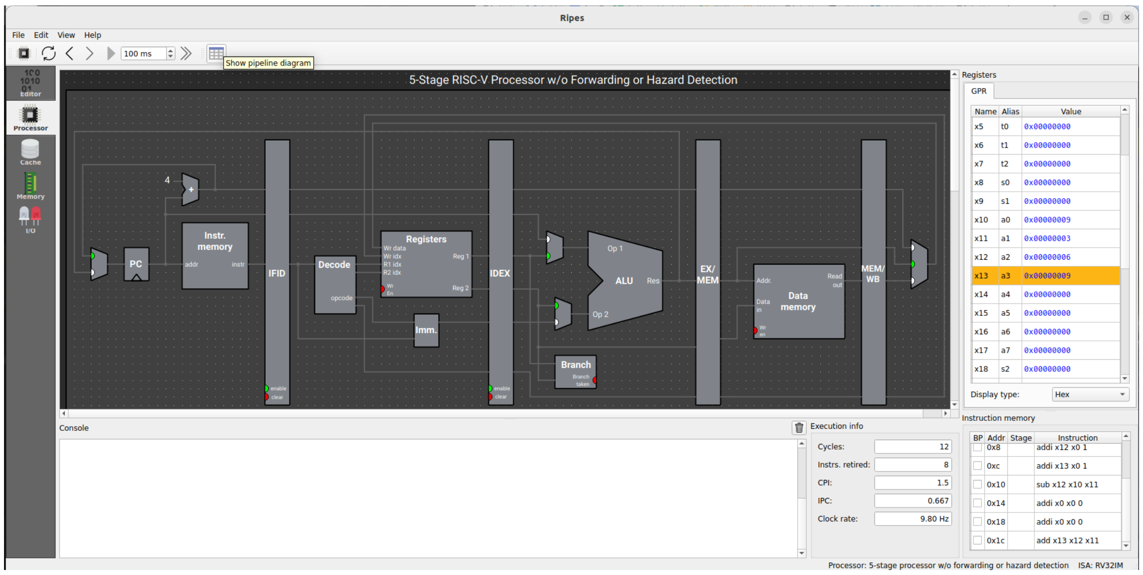
2 Q2

2.1 a. Read after write hazard

- 'nop' operations are introduced into code
- Explanation: 'nop' operations give cycles to complete writing into a2 and after completion it will read from a2. So updated value of a2 is used which gives correct output.
- Expected output = Obtained output = 9

```
.text
main:
    li a0, 9         # Initializing a0 = 9
    li a1, 3         # Initializing a1 = 3
    li a2, 1         # Initializing a2 = 1
    li a3, 1         # Initializing a3 = 1
    sub a2, a0, a1   # subtracting a1 from a0
    nop
    nop
    # value in a2 becomes 6(=9-3) after adding 2 nop operations
    add a3, a2, a1   # adding a2 and a1 and storing it into a3
    # Value in a3 becomes 9(=6+3)
```

Listing 2: Code for q2-a



Registers — GPR

Name	Alias	Value
x5	t0	0x00000000
x6	t1	0x00000000
x7	t2	0x00000000
x8	s0	0x00000000
x9	s1	0x00000000
x10	a0	0x00000009
x11	a1	0x00000003
x12	a2	0x00000006
x13	a3	0x00000009
x14	a4	0x00000000
x15	a5	0x00000000
x16	a6	0x00000000
x17	a7	0x00000000
x18	s2	0x00000000

Display type: Hex

Execution info

Cycles:	12
Instrs. retired:	8
CPI:	1.5
IPC:	0.667
Clock rate:	9.80 Hz

Instruction memory

BP	Addr	Stage	Instruction
	0x8		addi x12 x0 1
	0xc		addi x13 x0 1
	0x10		sub x12 x10 x11
	0x14		addi x0 x0 0
	0x18		addi x0 x0 0
	0x1c		add x13 x12 x11

Processor: 5-stage processor w/o forwarding or hazard detection ISA: RV32IM

Figure 2: ’nop’ instruction is used to eliminate all hazards

2.2 b. Write after read hazard

Not applicable

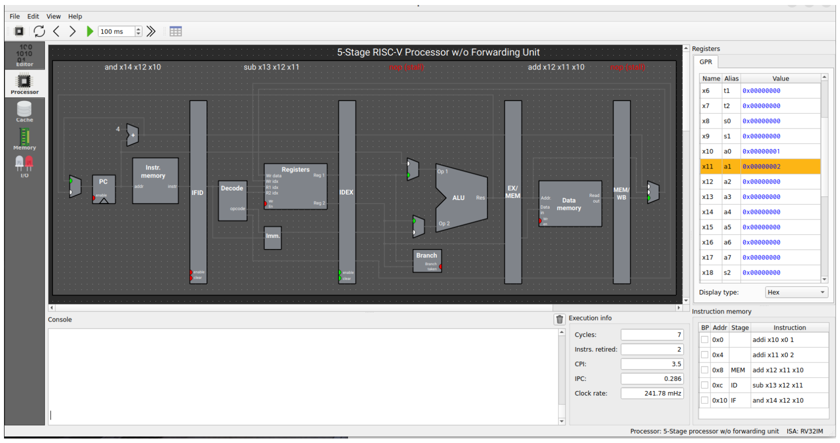
3 Q3

3.1 3a

- Writing into register and immediately using that same register to read it's value can change the number of cycles used in 2 mentioned processors(5 stage processor without forwarding and 5 stage processor).
- Without forwarding, the registers don't get updated (for writing) in 1 cycle so it takes more cycle to update value and then read the value from register. With forwarding as soon as answer is calculated it is sent in next step so it uses less number of cycles than in without forwarding.
- No. of cycles without forwarding:13, No. of cycles with forwarding:9

```
.text
main:
    li a0, 1     # Initializing a0 to 1
    li a1, 2     # Initializing a1 to 2
# Storing addition of a0 and a1 into a2
    add a2, a1, a0
# Reading from a2
    sub a3, a2, a1
    and a4, a2, a0
```

Listing 3: Code for q3-a



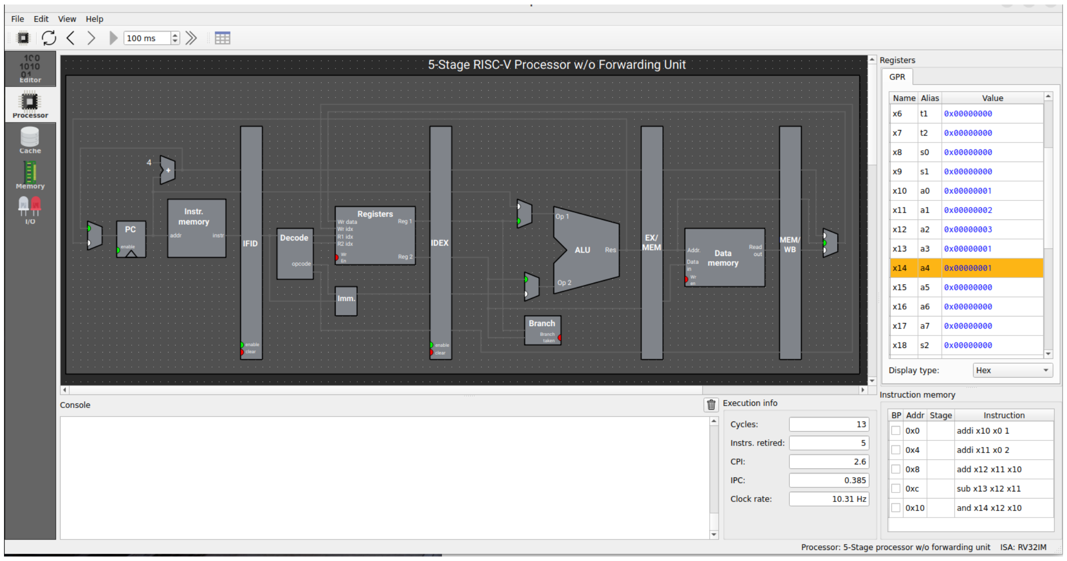
Name	Alias	Value
x6	t1	0x00000000
x7	t2	0x00000000
x8	s0	0x00000000
x9	s1	0x00000000
x10	a0	0x00000001
x11	a1	0x00000002
x12	a2	0x00000000
x13	a3	0x00000000
x14	a4	0x00000000
x15	a5	0x00000000
x16	a6	0x00000000
x17	a7	0x00000000
x18	s2	0x00000000

Cycles:	7
Instrs. retired:	2
CPI:	3.5
IPC:	0.286
Clock rate:	241.78 mHz

BP	Addr	Stage	Instruction
☐	0x0		addi x10 x0 1
☐	0x4		addi x11 x0 2
☐	0x8	MEM	add x12 x11 x10
☐	0xc	ID	sub x13 x12 x11
☐	0x10	IF	and x14 x12 x10

Processor: 5-Stage processor w/o forwarding unit   ISA: RV32IM

Figure 3: Stalls in the execution



Registers

GPR

Name	Alias	Value
x6	t1	0x00000000
x7	t2	0x00000000
x8	s0	0x00000000
x9	s1	0x00000000
x10	a0	0x00000001
x11	a1	0x00000002
x12	a2	0x00000003
x13	a3	0x00000001
x14	a4	0x00000001
x15	a5	0x00000000
x16	a6	0x00000000
x17	a7	0x00000000
x18	s2	0x00000000

Display type: Hex

Execution info

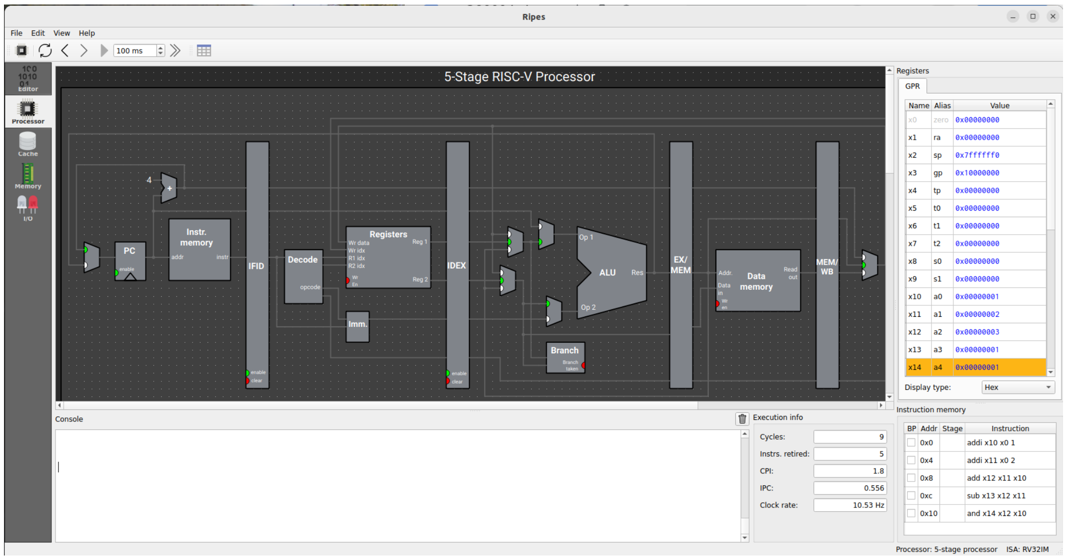
Cycles:	13
Instrs. retired:	5
CPI:	2.6
IPC:	0.385
Clock rate:	10.31 Hz

Instruction memory

BP	Addr	Stage	Instruction
☐	0x0		addi x10 x0 1
☐	0x4		addi x11 x0 2
☐	0x8		add x12 x11 x10
☐	0xc		sub x13 x12 x11
☐	0x10		and x14 x12 x10

Processor: 5-Stage processor w/o forwarding unit ISA: RV32IM

Figure 4: Using 5 stage processor without forwarding



Registers

GPR

Name	Alias	Value
x0	zero	0x00000000
x1	ra	0x00000000
x2	sp	0x7ffffff0
x3	gp	0x10000000
x4	tp	0x00000000
x5	t0	0x00000000
x6	t1	0x00000000
x7	t2	0x00000000
x8	s0	0x00000000
x9	s1	0x00000000
x10	a0	0x00000001
x11	a1	0x00000002
x12	a2	0x00000003
x13	a3	0x00000001
x14	a4	0x00000001

Display type: Hex

Execution info

Cycles:	9
Instrs. retired:	5
CPI:	1.8
IPC:	0.556
Clock rate:	10.53 Hz

Instruction memory

BP	Addr	Stage	Instruction
☐	0x0		addi x10 x0 1
☐	0x4		addi x11 x0 2
☐	0x8		add x12 x11 x10
☐	0xc		sub x13 x12 x11
☐	0x10		and x14 x12 x10

Processor: 5-stage processor ISA: RV32IM

Figure 5: Using 5 stage processor

3.2 3b

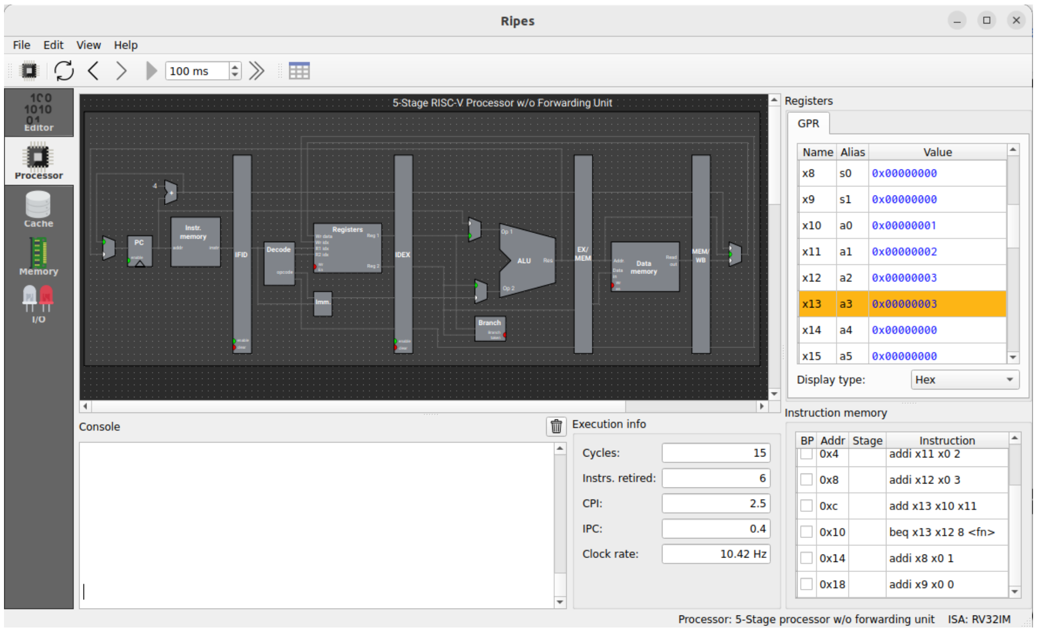
- See line number 6 and 7 in algorithm 4 and 5 they are swapped. There swapping gives different number of cycles.
- In q3-b_original, 'beq' command (line 9) has to wait for more cycles because writing into register a3 then reading from it takes more cycles as there is no forwarding. In q3-b_modified, as writing into register a3 and reading from it separated from other command so there are no requirements of extra cycles (The (li a2, 3) is done into extra cycle time so there is no time loss)
- No. of cycles in q3-b_original=15, No. of cycles in q3-b_modified= 16

```
 1  .text
 2  main:
 3      li a0, 1    # Initializing a0 to 1
 4      li a1, 2    # Initializing a0 to 2
 5      # Next 2 instructions are swapped
 6      li a2, 3    # Initializing a0 to 3
 7      add a3, a0, a1    # Adding a0 and a1
 8      beq a3, a2, fn
 9      li s0, 1
10  fn:
11      li s1, 0
```

Listing 4: Code for q3-b_original

```
 1  .text
 2  main:
 3      li a0, 1    # Initializing a0 to 1
 4      li a1, 2    # Initializing a0 to 2
 5      # Next 2 instructions are swapped
 6      add a3, a0, a1    # Adding a0 and a1
 7      li a2, 3    # Initializing a0 to 3
 8      beq a3, a2, fn
 9      li s0, 1
10  fn:
11      li s1, 0
```

Listing 5: Code for q3-b_modified



Registers

GPR

Name	Alias	Value
x8	s0	0x00000000
x9	s1	0x00000000
x10	a0	0x00000001
x11	a1	0x00000002
x12	a2	0x00000003
x13	a3	0x00000003
x14	a4	0x00000000
x15	a5	0x00000000

Display type: Hex

Execution info

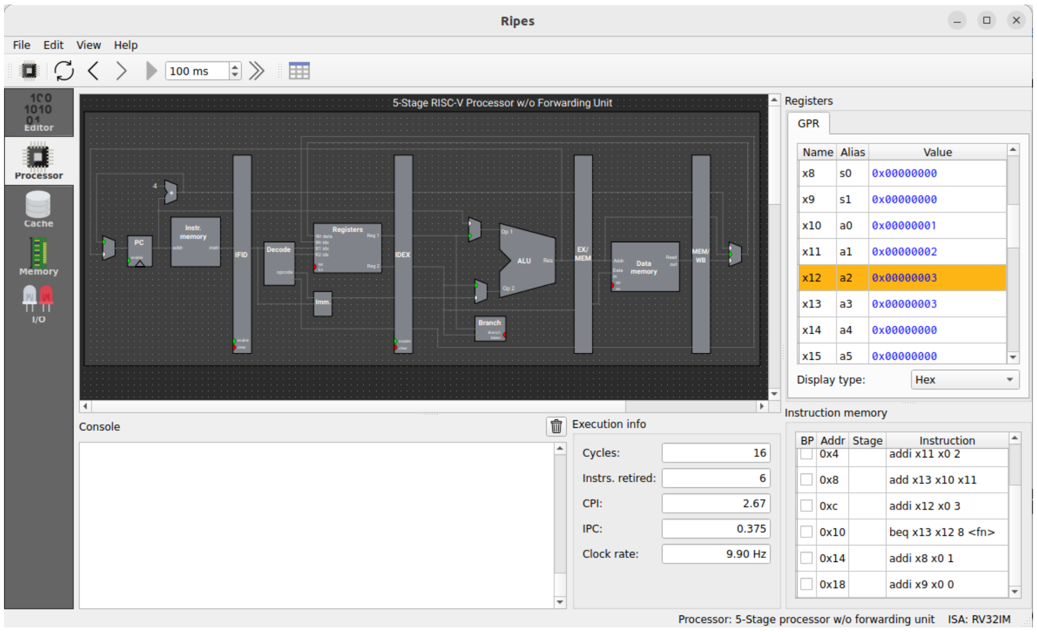
Cycles:	15
Instrs. retired:	6
CPI:	2.5
IPC:	0.4
Clock rate:	10.42 Hz

Instruction memory

BP	Addr	Stage	Instruction
	0x4		addi x11 x0 2
	0x8		addi x12 x0 3
	0xc		add x13 x10 x11
	0x10		beq x13 x12 8 <fn>
	0x14		addi x8 x0 1
	0x18		addi x9 x0 0

Processor: 5-Stage processor w/o forwarding unit ISA: RV32IM

Figure 6: Execution of original code



Registers

GPR

Name	Alias	Value
x8	s0	0x00000000
x9	s1	0x00000000
x10	a0	0x00000001
x11	a1	0x00000002
x12	a2	0x00000003
x13	a3	0x00000003
x14	a4	0x00000000
x15	a5	0x00000000

Display type: Hex

Execution info

Cycles:	16
Instrs. retired:	6
CPI:	2.67
IPC:	0.375
Clock rate:	9.90 Hz

Instruction memory

BP	Addr	Stage	Instruction
	0x4		addi x11 x0 2
	0x8		add x13 x10 x11
	0xc		addi x12 x0 3
	0x10		beq x13 x12 8 <fn>
	0x14		addi x8 x0 1
	0x18		addi x9 x0 0

Processor: 5-Stage processor w/o forwarding unit ISA: RV32IM

Figure 7: Execution of code after changing order of instructions

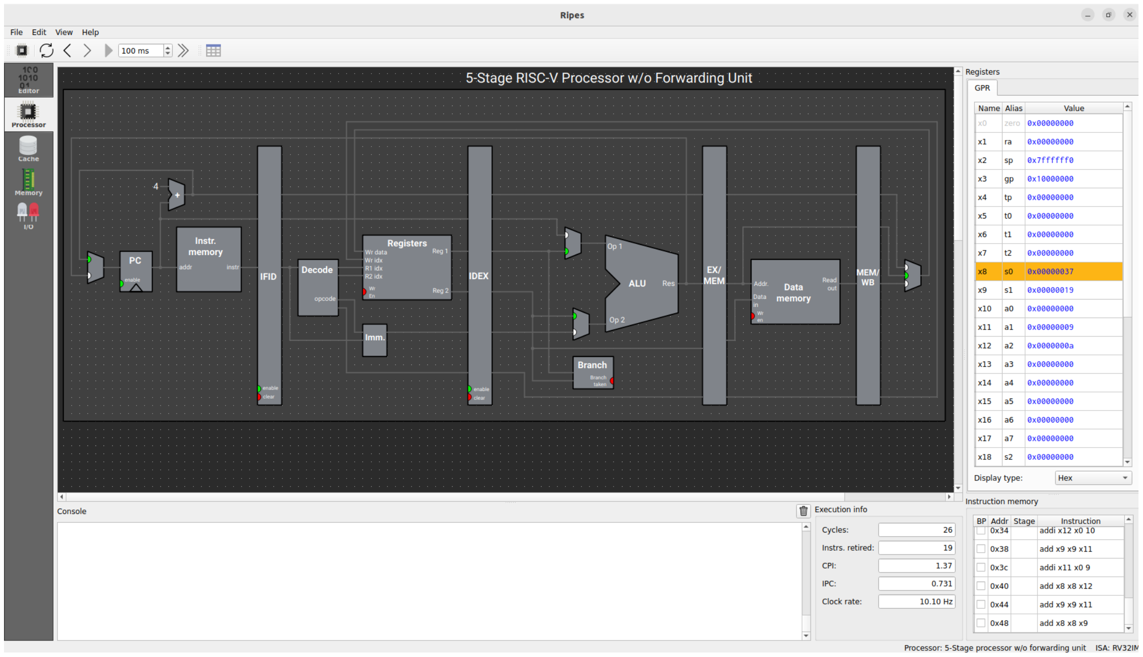
4 Q4

- Number of cycles : 26
- Number of registers: 4 (s0, s1, a1, a2)
- As forwarding is absent there should be two different instructions in between the operations on 1 register. So I splitted sum into half sum of even and sum of odds and added them in last.

```
.text
main:
    # s0 stores sum of even numbers {2, 4, 6, 8, 10}
    li s0, 2
    # a2 is used to load even numbers
    li a2, 4
    # s1 stores sum of odd numbers {1, 3, 5, 7, 9}
    li s1, 1
    # a1 is used to load odd numbers
    li a1, 3
    add s0, s0, a2
    li a2, 6
    add s1, s1, a1
    li a1, 5
    add s0, s0, a2
    li a2, 8
    add s1, s1, a1
    li a1, 7
    add s0, s0, a2
    li a2, 10
    add s1, s1, a1
    li a1, 9
    add s0, s0, a2
    add s1, s1, a1
    # Final addition stored in s0
    add s0, s0, s1

```

Listing 6: Code for q4



Registers

GPR

Name	Alias	Value
x0	zero	0x00000000
x1	ra	0x00000000
x2	sp	0x7ffffff0
x3	gp	0x10000000
x4	tp	0x00000000
x5	t0	0x00000000
x6	t1	0x00000000
x7	t2	0x00000000
x8	s0	0x00000037
x9	s1	0x00000019
x10	a0	0x00000000
x11	a1	0x00000009
x12	a2	0x0000000a
x13	a3	0x00000000
x14	a4	0x00000000
x15	a5	0x00000000
x16	a6	0x00000000
x17	a7	0x00000000
x18	s2	0x00000000

Display type: Hex

Execution info

Cycles:	26
Instrs. retired:	19
CPI:	1.37
IPC:	0.731
Clock rate:	10.10 Hz

Instruction memory

BP	Addr	Stage	Instruction
	0x34		addi x12 x0 10
	0x38		add x9 x9 x11
	0x3c		addi x11 x0 9
	0x40		add x8 x8 x12
	0x44		add x9 x9 x11
	0x48		add x8 x8 x9

Processor: 5-Stage processor w/o forwarding unit ISA: RV32IM

Figure 8: Q4