ASSIGNMENT - 3

ASSIGNMENT - 3

1.1. Provide the instruction type and assembly language instruction for the following binary value

0000000	00001	00001	000	00001	0110011
Func 7	rs2	rs1	Func3	rd	opcode

→

Func7 = 0x0

rs 2 = $ra

rs 1 = $ra

Func 3 = 0x0

rd = $ra

opcode = 011 0011

The instruction type is R-Type.

Assembly level instructions:-

ADD ra, ra, ra ...... Instruction Format:- ADD rd, rs1, rs2

1.2 Provide the instruction type and hex and binary representation of the following instructions

sw x5, 32 (x30)

→ sw opcode = 35 = 0100011 Instruction type is S-type.

imm2 7 bits	rs2 5 bits	rs1 5 bits	func3 3 bits	imm1 5 bits	opcode 7 bits
0000001	00101	11110	010	00000	0100011

0 2 5 F 2 0 2 3

Binary representation

Hex representation = 025F2023

2. Follow the factorial( ) example in lecture, implement sum( ) function in RISC-C assembly instructions.
For example, sum(5) will calculate 1 + 2 + 3 + 4 + 5
Use online RISC-V simulator to run your program
Some of the online simulators are:

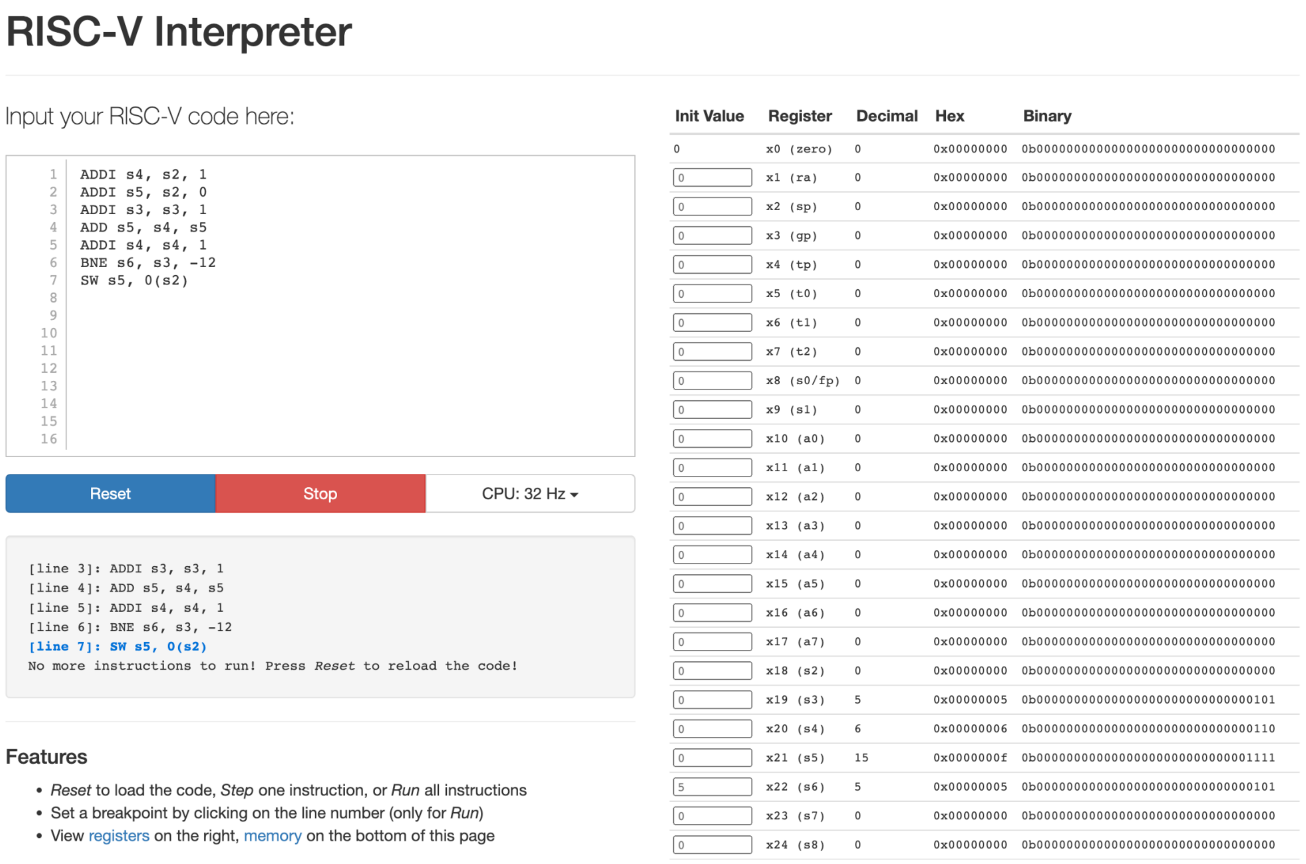
1. https://www.cs.cornell.edu/courses/cs3410/2019sp/riscv/interpreter/
2. https://riscvasm.lucasteske.dev/
3. https://ascslab.org/research/briscv/simulator/simulator.html

Submit code and running screen shot.

Solution:

ADDI s4, s2, 1
ADDI s5, s2, 0
ADDI s3, s3, 1
ADD s5, s4, s5
ADDI s4, s4, 1
BNE s6, s3, -12
SW s5, 0(s2)

Screenshot:



RISC-V Interpreter

Input your RISC-V code here:

```
 1  ADDI s4, s2, 1
 2  ADDI s5, s2, 0
 3  ADDI s3, s3, 1
 4  ADD s5, s4, s5
 5  ADDI s4, s4, 1
 6  BNE s6, s3, -12
 7  SW s5, 0(s2)
 8
 9
10
11
12
13
14
15
16
```

```
[line 3]: ADDI s3, s3, 1
[line 4]: ADD s5, s4, s5
[line 5]: ADDI s4, s4, 1
[line 6]: BNE s6, s3, -12
[line 7]: SW s5, 0(s2)
No more instructions to run! Press Reset to reload the code!
```

Init Value	Register	Decimal	Hex	Binary
0	x0 (zero)	0	0x00000000	0b00000000000000000000000000000000
0	x1 (ra)	0	0x00000000	0b00000000000000000000000000000000
0	x2 (sp)	0	0x00000000	0b00000000000000000000000000000000
0	x3 (gp)	0	0x00000000	0b00000000000000000000000000000000
0	x4 (tp)	0	0x00000000	0b00000000000000000000000000000000
0	x5 (t0)	0	0x00000000	0b00000000000000000000000000000000
0	x6 (t1)	0	0x00000000	0b00000000000000000000000000000000
0	x7 (t2)	0	0x00000000	0b00000000000000000000000000000000
0	x8 (s0/fp)	0	0x00000000	0b00000000000000000000000000000000
0	x9 (s1)	0	0x00000000	0b00000000000000000000000000000000
0	x10 (a0)	0	0x00000000	0b00000000000000000000000000000000
0	x11 (a1)	0	0x00000000	0b00000000000000000000000000000000
0	x12 (a2)	0	0x00000000	0b00000000000000000000000000000000
0	x13 (a3)	0	0x00000000	0b00000000000000000000000000000000
0	x14 (a4)	0	0x00000000	0b00000000000000000000000000000000
0	x15 (a5)	0	0x00000000	0b00000000000000000000000000000000
0	x16 (a6)	0	0x00000000	0b00000000000000000000000000000000
0	x17 (a7)	0	0x00000000	0b00000000000000000000000000000000
0	x18 (s2)	0	0x00000000	0b00000000000000000000000000000000
0	x19 (s3)	5	0x00000005	0b00000000000000000000000000000101
0	x20 (s4)	6	0x00000006	0b00000000000000000000000000000110
0	x21 (s5)	15	0x0000000f	0b00000000000000000000000000001111
5	x22 (s6)	5	0x00000005	0b00000000000000000000000000000101
0	x23 (s7)	0	0x00000000	0b00000000000000000000000000000000
0	x24 (s8)	0	0x00000000	0b00000000000000000000000000000000

Features

- *Reset* to load the code, *Step* one instruction, or *Run* all instructions
- Set a breakpoint by clicking on the line number (only for *Run*)
- View registers on the right, memory on the bottom of this page

Memory Address 0x00000000 Go Download!

Memory Address	Decimal	Hex	Binary
0x00000000	15	0x0000000f	0b00000000000000000000000000001111
0x00000004	0	0x00000000	0b00000000000000000000000000000000
0x00000008	0	0x00000000	0b00000000000000000000000000000000
0x0000000c	0	0x00000000	0b00000000000000000000000000000000
0x00000010	0	0x00000000	0b00000000000000000000000000000000
0x00000014	0	0x00000000	0b00000000000000000000000000000000
0x00000018	0	0x00000000	0b00000000000000000000000000000000
0x0000001c	0	0x00000000	0b00000000000000000000000000000000
0x00000020	0	0x00000000	0b00000000000000000000000000000000
0x00000024	0	0x00000000	0b00000000000000000000000000000000