Computer Organization Project

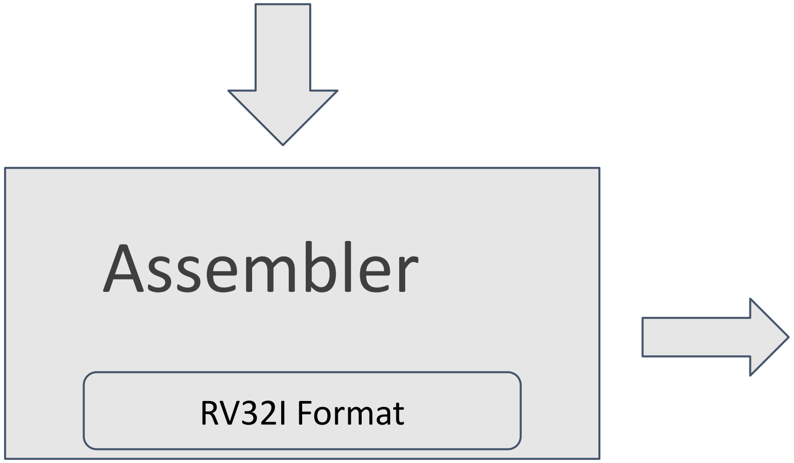
Part I: Assembler

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Assembler Task

```
addi    a0,zero,-5
addi    a1,zero,3
sltiu   t0,a0,-1
sltiu   t1,a1,2
sll t2,a0,a1
sub a0,a0,a1
slt t3,zero,a0
beq zero,zero,0
```

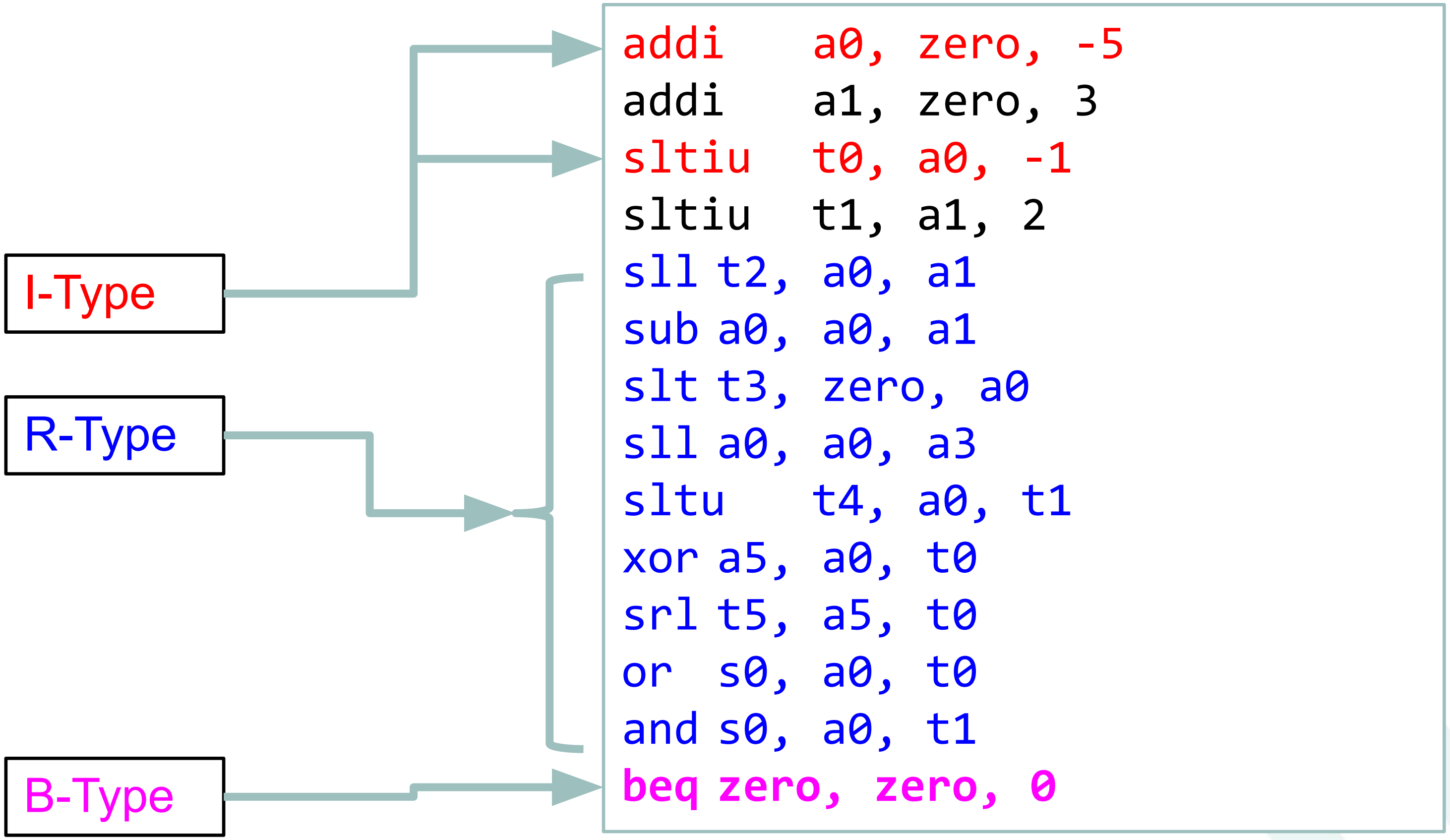
< Input Assembly code file



```
11111111101100000000010100010011
00000000001100000000010110010011
11111111111101010011001010010011
00000000001001011011001100010011
00000000101101010001001110110011
01000000101101010000010100110011
00000000101000000010111000110011
00000000000000000000000001100011
```

Output Binary Code File

Instruction Types



I-Type

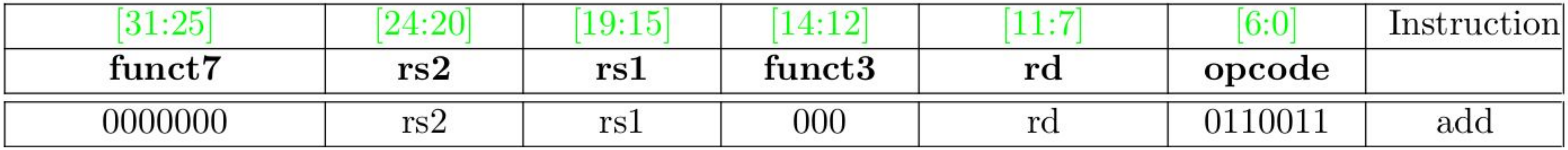
R-Type

B-Type

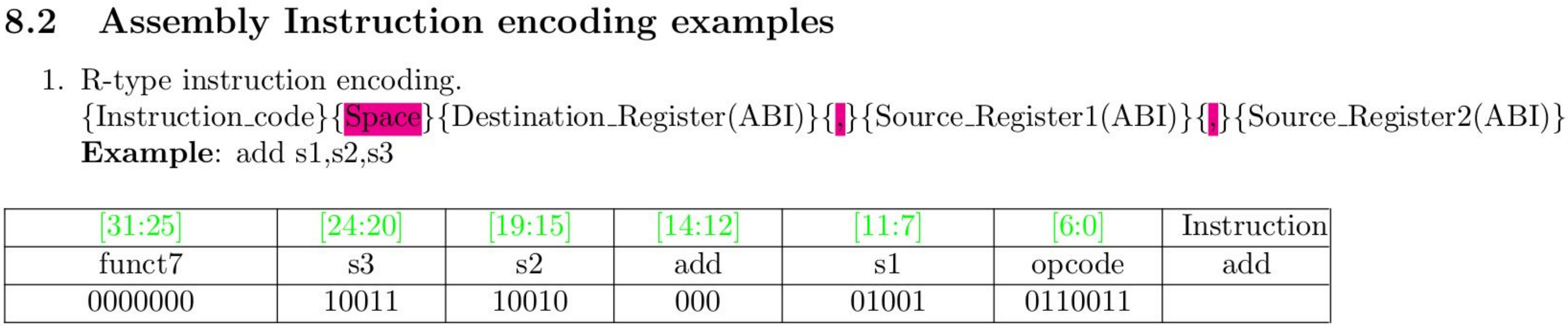
```
addi    a0, zero, -5
addi    a1, zero, 3
sltiu   t0, a0, -1
sltiu   t1, a1, 2
sll t2, a0, a1
sub a0, a0, a1
slt t3, zero, a0
sll a0, a0, a3
sltu    t4, a0, t1
xor a5, a0, t0
srl t5, a5, t0
or  s0, a0, t0
and s0, a0, t1
beq zero, zero, 0
```

Instruction Encoding

R-Type Instruction



[31:25]	[24:20]	[19:15]	[14:12]	[11:7]	[6:0]	Instruction
funct7	**rs2**	**rs1**	**funct3**	**rd**	**opcode**	
0000000	rs2	rs1	000	rd	0110011	add



8.2 Assembly Instruction encoding examples

1. R-type instruction encoding.
   {Instruction_code}{Space}{Destination_Register(ABI)}{,}{Source_Register1(ABI)}{,}{Source_Register2(ABI)}
   Example: add s1,s2,s3

[31:25]	[24:20]	[19:15]	[14:12]	[11:7]	[6:0]	Instruction
funct7	s3	s2	add	s1	opcode	add
0000000	10011	10010	000	01001	0110011	

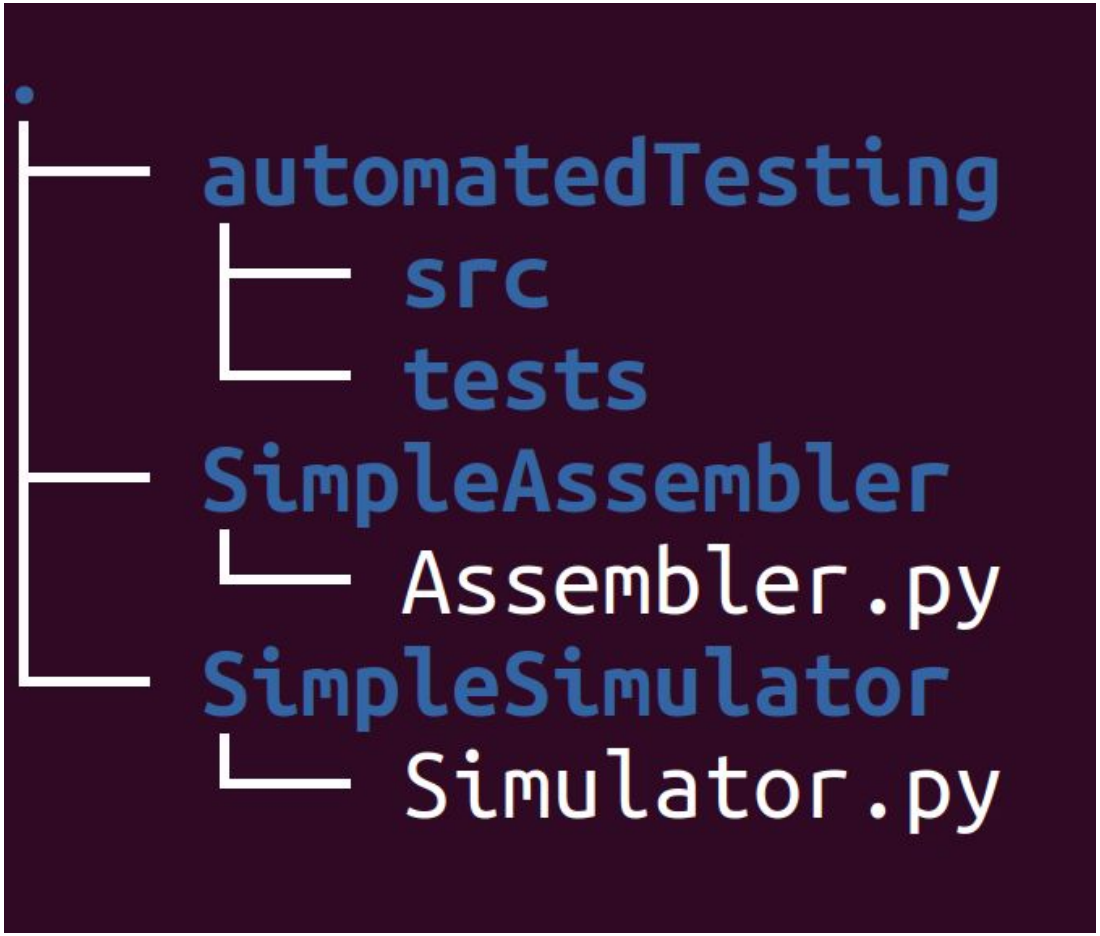
Label in Instruction

Address	**Instruction**	
0x00	**start:** add ra,sp,gp	add ra,sp,gp
0x04	jalr t0,t1,4	jalr t0,t1,4
0x08	beq ra,sp,start	beq ra,sp,-8
0x0C	beq zero,zero,0	beq zero,zero,0

Note on "label":

1. start is initialized with Instruction Address =0x00
2. start is decoded as start - current PC. And, keep in decimal format.
   a. decode result = 0x00 - 0x08
3. If label is negative convert into 2's complement format.

Assembler Evaluation



```
.
├── automatedTesting
│   ├── src
│   └── tests
├── SimpleAssembler
│   └── Assembler.py
└── SimpleSimulator
    └── Simulator.py
```

Pink arrow indicate where you should put assembler and simulator

Assembler Evaluation

Step 1: Put your assembler in

```
/evaluation_framework/SimpleAssembler/Assembler.py
```

Step 2 : Rename your assembler code file as "*Assembler.py*"

Step 3 : Go to :

```
evaluation_framework/automatedTesting
```

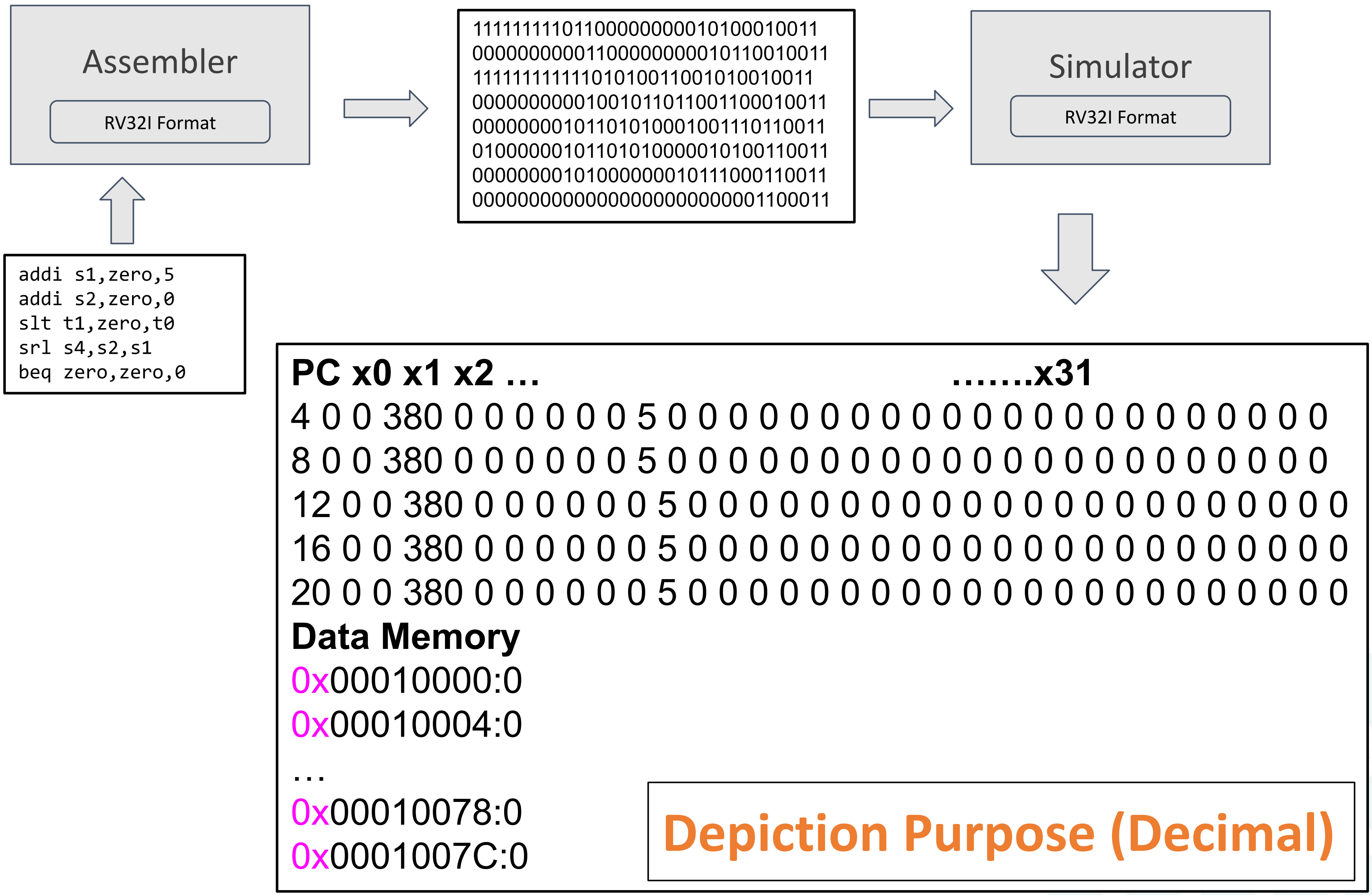
Step 4 :

linux users: $ `python3 src/main.py --no-sim --linux`

windows user: > `python3 src\main.py --no-sim --windows`

Note: Refer to readme file in the project description.

Assignment Complete Overview



```
addi s1,zero,5
addi s2,zero,0
slt t1,zero,t0
srl s4,s2,s1
beq zero,zero,0
```

```
11111111101100000000010100010011
00000000001100000000010110010011
11111111111101010011001010010011
00000000001001011011001100010011
00000000101101010001001110110011
01000000101101010000010100110011
00000000101000000010111000110011
00000000000000000000000001100011
```

```
PC x0 x1 x2 …                              …….x31
4 0 0 380 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
8 0 0 380 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
12 0 0 380 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
16 0 0 380 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
20 0 0 380 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Data Memory
0x00010000:0
0x00010004:0
…
0x00010078:0
0x0001007C:0
```

Depiction Purpose (Decimal)

Simulator Output Rectification

~~PC x0 x1 x2 … …….x31~~

4 0 0 380 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

⇩

binary(4) **binary**(0) **binary**(0) **binary**(380) ……………**binary**(0)

⇩

0b00000000000000000000000000000100 …. **binary**(0)

~~Data Memory~~

0x00010000:**binary**(0)

…

0x0001007C:**binary**(0)

Note: The starting memory address is the specified data memory address.

⇩

0x00010000:0b00000000000000000000000000000000

…

0x0001007C:0b00000000000000000000000000000000

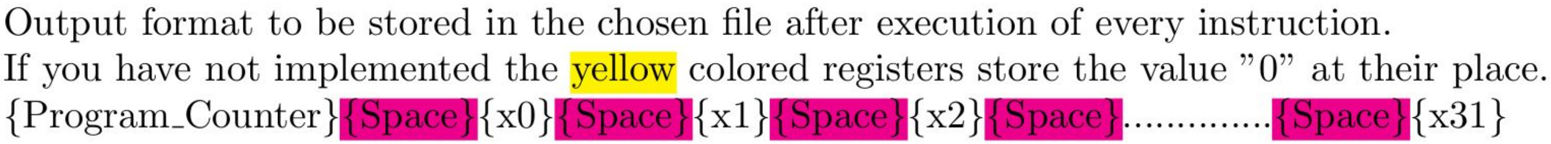
Simulator Output Rectification

continued

~~PC x0 x1 x2 … …….x31~~

4 0 0 380 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0b00000000000000000000000000000100 …. **binary**(0)

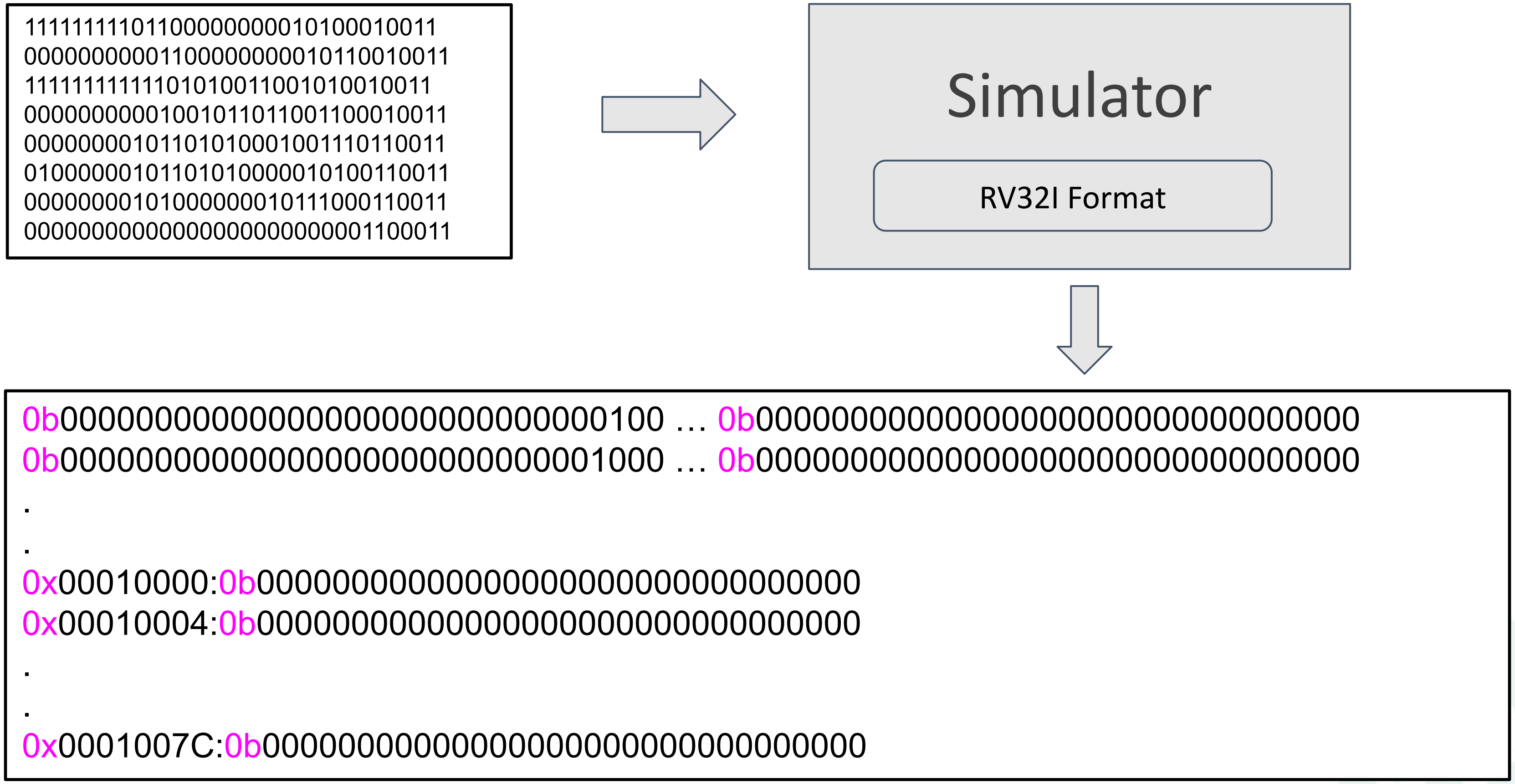


Output format to be stored in the chosen file after execution of every instruction.
If you have not implemented the yellow colored registers store the value ”0” at their place.
{Program_Counter}{Space}{x0}{Space}{x1}{Space}{x2}{Space}............{Space}{x31}

Note: Negative numbers are stored in 2’s complement form.

Simulator Task

Input Machine code file



```
11111111101100000000010100010011
00000000001100000000010110010011
11111111111101010011001010010011
00000000001001011011001100010011
00000000101101010001001110110011
01000000101101010000010100110011
00000000101000000010111000110011
00000000000000000000000001100011
```

Simulator

RV32I Format

```
0b00000000000000000000000000000100 … 0b00000000000000000000000000000000
0b00000000000000000000000000001000 … 0b00000000000000000000000000000000
.
.
0x00010000:0b00000000000000000000000000000000
0x00010004:0b00000000000000000000000000000000
.
.
0x0001007C:0b00000000000000000000000000000000
```

Output Simulation Trace File

Simulator Task Guidelines

- After every instruction execution (including HALT); store each register's value (including PC).
- Negative values are stored in 32-bit 2's complement form.
- The address of data memory are stored in hex format and the data in binary format.
- Exit the Simulator after execution of first HALT instruction.
- Unitil explicitly specified. Assume all registers (including PC) are initialized with decimal zero.
- The starting and ending memory address are the specified data memory addresses.
- The instruction trace will be at the top and the memory trace will be at the bottom.

Simulator Evaluation

Step 1: Put your simulator in

```
/evaluation_framework/SimpleSimulator/Simulator.py
```

Step 2 : Rename your simulator code file as "*Simulator.py*"

Step 3 : Go to :

```
evaluation_framework/automatedTesting
```

Step 4 :

linux users: $ `python3 src/main.py --no-asm --linux`

windows user: > `python3 src\main.py --no-asm --windows`

Note: Refer to readme file in the project description.

Where to Raise query

Project TA
Naorem Akshaykumar
Keshav Goel
Ajinkya Ghawale
Priyanshu Kumar Rai
Vishal Kumar
Sameena Khan
Anushree Vardish
Manshaa Kapoor
Ekansh

Thanks