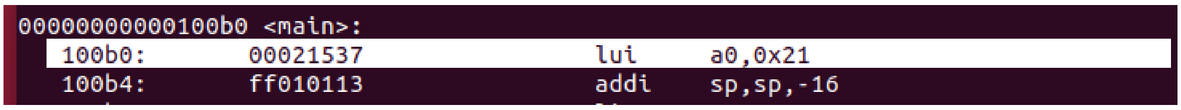
RISC-V Talent Development Program
Excellent Opportunity for Engineering Students
Kickstarting in January 2025
SAMSUNG
POWERED BY
Samsung Semiconductor India Research
LEARNING PARTNERS
AGASTYA

TASK 3

15 INSTRUCTIONS

1. Instruction –

lui a0, 0*21

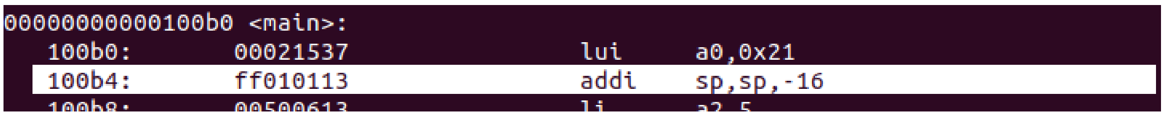


```
00000000000100b0 <main>:
   100b0:       00021537                lui     a0,0x21
   100b4:       ff010113                addi    sp,sp,-16
```

Type: U-Type
Opcode: 0110111 (lui)
rd: a0 = x10 → 01010
Immediate: 0x21 → 0000000000100001 (20 bits)
Overall:
imm[31:12] | rd | opcode
0000000000100001 | 01010 | 0110111

2. Instruction –

addi sp, sp, -16

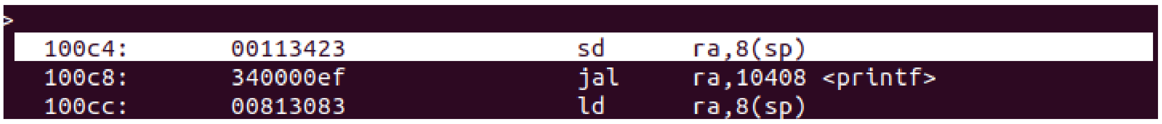


```
00000000000100b0 <main>:
   100b0:       00021537                lui     a0,0x21
   100b4:       ff010113                addi    sp,sp,-16
   100b8:       00500613                li      a2,5
```

Type: I-Type
Opcode: 0010011 (addi)
funct3: 000
rd: sp = x2 → 00010
rs1: sp = x2 → 00010
Immediate: -16 → 1111111111110000 (12-bit two’s complement)
Overall:
imm[11:0] | rs1 | funct3 | rd | opcode
111111111000 | 00010 | 000 | 00010 | 0010011

3. Instruction –

sd ra, 8(sp)

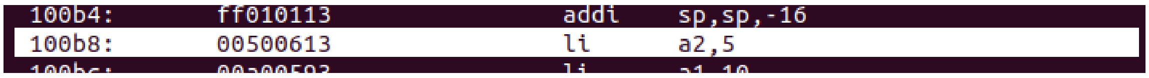


```
100c4:        00113423                sd      ra,8(sp)
100c8:        340000ef                jal     ra,10408 <printf>
100cc:        00813083                ld      ra,8(sp)
```

Type: S-Type
Opcode: 0100011 (sd)
funct3: 011
rs1: sp = x2 → 00010
rs2: ra = x1 → 00001
Immediate: 8 → 0000000000001000 (split as imm[11:5] and imm[4:0])
Overall:
imm[11:5] | rs2 | rs1 | funct3 | imm[4:0] | opcode
0000000 | 00001 | 00010 | 011 | 01000 | 0100011

4. Instruction –

li a2, 5



```
100b4:        ff010113                addi    sp,sp,-16
100b8:        00500613                li      a2,5
```

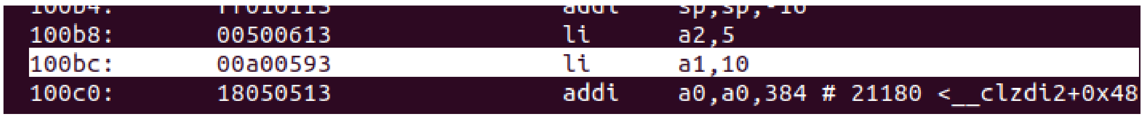
- li a2, 5 is a **pseudo-instruction**.
- It is **not directly encoded** in RISC-V machine code. Instead, it is implemented as the equivalent **real instruction**:
  addi a2, x0, 5

Instruction: Add immediate (addi)
Destination Register (rd): a2 (register x12)(01100).
Source Register (rs1): x0 (always zero)(00000).
Immediate Value: 5 (000000000101)
funct3:For addi, funct3 = 000

imm[11:0]	**rs1**	**funct3**	**rd**	**opcode**
000000000101	00000	000	01100	0010011

5. Instruction

li a1, 10



```
100b8:        00500613                li      a2,5
100bc:        00a00593                li      a1,10
100c0:        18050513                addi    a0,a0,384 # 21180 <__clzdi2+0x48
```

- li a1, 10 is a pseudo-instruction in RISC-V.
- It is implemented as the real instruction:
  addi a1, x0, 10

Immediate Field:
- 10 in decimal is 000000001010 in **12-bit binary**.

Source Register (rs1):
- x0 is the source register: 00000.

Destination Register (rd):
- a1 corresponds to x11: 01011 in binary.

funct3:
- For addi, funct3 is 000.

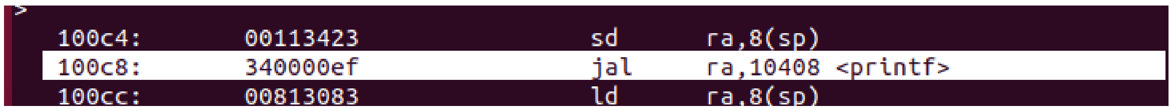
Opcode:

- The opcode for addi is 0010011.

imm[11:0]	rs1	funct3	rd	opcode
000000001010	00000	000	01011	0010011

6. Instruction –

Jal ra, 104088<printf>



```
>
100c4:          00113423                sd      ra,8(sp)
100c8:          340000ef                jal     ra,10408 <printf>
100cc:          00813083                ld      ra,8(sp)
```

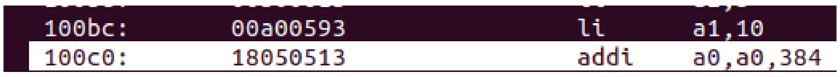
- It computes the relative offset to 10408 and stores the return address in ra**.**

The jal instruction format is:

imm[20]	imm[10:1]	imm[11]	imm[19:12]	rd	opcode
0	**0110011110**	**0**	**00000000**	**00001**	**1101111**

7. Instruction –

Addi a0,



```
100bc:          00a00593                li      a1,10
100c0:          18050513                addi    a0,a0,384
```

- adds the **immediate value** 384 to the value in register a0 and stores the result in the same register a0.
- addi uses the **I-Type format**.

Immediate (imm) 12 bits 84

Source Register (rs1) 5 bits a0 (x10)

Destination Register (rd) 5 bits a0 (x10)

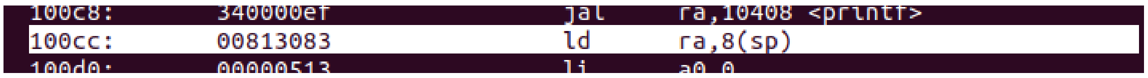
funct3 3 bits 000 (for addi)

Opcode 7 bits 0010011

imm[11:0]	rs1	funct3	rd	opcode
000000110000	01010	000	01010	0010011

8. Instruction –

ld ra, 8(sp)



```
100c8:          340000ef                jal     ra,10408 <printf>
100cc:          00813083                ld      ra,8(sp)
100d0:          00000513                li      a0,0
```

Immediate (imm) 12 bits 8 (offset value)

Source Register (rs1) 5 bits sp (x2)

Destination Register (rd) 5 bits ra (x1)

funct3 3 bits 011 (for ld)

Opcode 7 bits **I-Type** instruction - 0000011

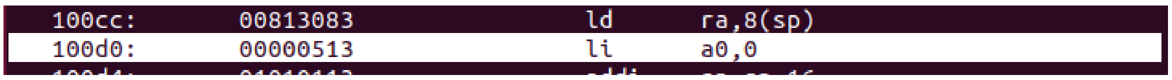
Immediate (imm)	12 bits	0 (value to load)

imm[11:0]	**rs1**	**funct3**	**rd**	**opcode**
000000000100	00010	011	00001	0000011

9. Instruction –

li a0, 0

- The **li (load immediate)** pseudo-instruction is equivalent to addi with an immediate value.
- It loads the value 0 into the register a0.
- In RISC-V, li a0, 0 translates to:
  addi a0, x0, 0



```
   100cc:        00813083                ld      ra,8(sp)
   100d0:        00000513                li      a0,0
```

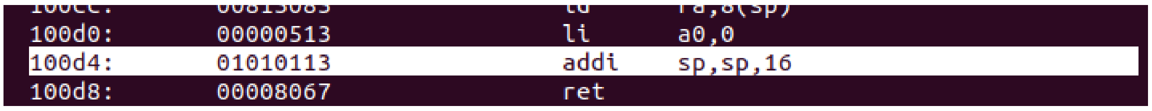
Source Register (rs1)	5 bits	x0 (always 0)
Destination Register (rd)	5 bits	a0 (x10)
funct3	3 bits	000 (for addi)
Opcode	7 bits	0010011

imm[11:0]	**rs1**	**funct3**	**rd**	**opcode**
000000000000	00000	000	01010	0010011

10.Instruction –

addi sp, sp, 16

- The **addi** instruction adds an immediate value (16) to the value in the source register (sp) and stores the result in the destination register (sp).
- **Instruction Type**: I-Type.



```
   100cc:        00813083                ld      ra,8(sp)
   100d0:        00000513                li      a0,0
   100d4:        01010113                addi    sp,sp,16
   100d8:        00008067                ret
```

Immediate (imm) 12 bits 16 (value to add)

Source Register (rs1) 5 bits sp (x2)

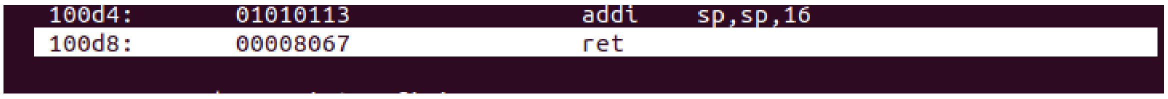
Destination Register (rd) 5 bits sp (x2)

funct3 3 bits 000 (for addi)

Opcode 7 bits 0010011

11.Instruction –

ret



```
   100d4:        01010113                addi    sp,sp,16
   100d8:        00008067                ret
```

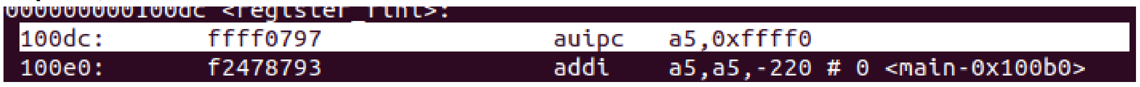
- **Instruction**: jalr x0, 0(ra)

- **Type**: I-Type
- **Opcode**: 1100111 (jalr)
- **funct3**: 000
- **rd**: x0 → 00000
- **rs1**: ra = x1 → 00001
- **Immediate**: 0 → 000000000000

imm[11:0] | rs1 | funct3 | rd | opcode
000000000000 | 00001 | 000 | 00000 | 1100111

12.Instruction

auipc a5, 0xffff0



```
0000000000100dc <register_fini>:
100dc:       ffff0797                 auipc   a5,0xffff0
100e0:       f2478793                 addi    a5,a5,-220 # 0 <main-0x100b0>
```

- The **auipc (Add Upper Immediate to PC)** instruction adds an immediate value shifted 12 bits left (<<12) to the current PC (program counter) and stores the result in the destination register a5.
- **Instruction Type**: U-Type.

Immediate (imm) 20 bits 0xffff0 (upper 20 bits)

Destination Register (rd) 5 bits a5 (x15)

Opcode 7 bits 0010111

imm[31:12]	**rd**	**opcode**
11111111111111110000	01111	0010111

13.Instruction –

addi a5, a0, 0 is:

Immediate	000000000000	12 bits	Immediate value 0 (zero).
Source Register (rs1)	01010 (x10, a0)	5 bits	Source register a0 (x10).
Funct3	000	3 bits	Encodes the addi operation.
Destination Register (rd)	01111 (x15, a5)	5 bits	Destination register a5 (x15).
Opcode	0010011	7 bits	Opcode for immediate arithmetic.

imm[11:0] | rs1 | funct3 | rd | opcode
000000000000 | 01010 | 000 | 01111 | 0010011

14.Instruction –

ld a0, -32(s0)
Immediate (imm):
- The offset is -32. In 12-bit two's complement:
  - -32 → 111111000000 (binary).

Source Register (rs1):

- s0 corresponds to x8 → 01000 (binary).

Destination Register (rd):

- a0 corresponds to x10 → 01010 (binary).

Function (funct3):

- Load doubleword (ld) → 011 (binary).

Opcode:

- Load instruction opcode → 0000011

imm[11:0]	**rs1**	**funct3**	**rd**	**opcode**
111111000000	01000	011	01010	0000011

15.Instruction-

addi a0, sp, 384
Type: I-Type
Opcode: 0010011 (addi)
funct3: 000
rd: a0 = x10 → 01010
rs1: sp = x2 → 00010
Immediate: 384 → 0000000110000000 (12-bit immediate)

```
imm[11:0]      | rs1     | funct3 | rd       | opcode
00001100000 | 00010 | 000     | 01010 | 0010011
```

SARVAMANGALA.B
RV INSTITUTE OF TECHNOLOGY AND MANAGEMENT