Chapter-2

Question - 1:

Construct the equivalent RISC-V code of the following C code. Once you have the RISC-V code, identify type of each instruction and encode them accordingly.

```
A[7] = A[2] + A[B[8]] + 10;
B[i] = A[3] - 8;
```

Base addresses of array A and B are in register X_{20} and X_{21} and i is in register X_{22}

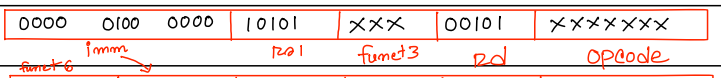
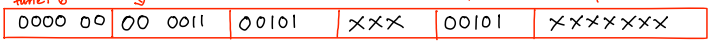
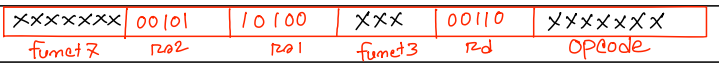
A = X_{20}
B = X_{21}
i = X_{22}

Line-1

```
ld   X5, 64(X21)     // B[8]
Slli X5, X5, 3       // offset calculation (X5 * 2^3)
Add  X6, X20, X5     // physical loc. (Base + offset)
ld   X6, 0(X6)       // A[B[8]]

ld   X5, 16(X20)     // A[2]

Add  X5, X5, X6      // A[B[8]] + A[2]

Addi X5, X5, 10      // A[B[8]] + A[2] + 10

Sd   X5, 56(X20)     // A[7] = A[B[8]] + A[2] + 10
```

Line-2

```
ld   X6, 24(X20)     // A[3]
Addi X6, X6, -8      // A[3] - 8

Slli X7, X22, 3      // offset calc => X22 * 2^3
Add  X7, X7, X21     // Physical Add. calc

Sd   X6, 0(X7)       // B[i] = A[3] - 8
```

No.	Instruction	Type	Encoding
1	ld x_5, 64(x_{21})	I ✓	
2	slli x_5, x_5, 3	I ✓	
3	Add x_6, x_{20}, x_5	R ✓	
4	ld x_6, 0(x_6)	I	
5	ld x_5, 16(x_{20})	I	
6	Add x_5, x_5, x_6	R	
7	Addi x_5, x_5, 10	I	
8	Sd x_5, 56(x_{20})	S ✓	
9	ld x_6, 18(x_{20})	I	
10	Addi x_6, x_6, -8	I ✓	
11	slli x_7, x_{22}, 3	I	
12	Add x_7, x_7, 21	R	
13	Sd x_6, 0(x_7)	S	

8 = 1000

+8 = 0 1000

+8 = 0000 0000 1000

1111 1111 0111

+1

-8 = 1111 1111 1000

Encode rest of the instructions yourself.

Question - 2:

Construct the equivalent RISC-V code of the following C code.

```
for (i = 8; i > 0 ; i--) {
  if ( A[i] == i){
     A[2] = A [B[3]] ;
  }
}
```

Base addresses of array A and B are in register X_{20} and X_{21} . Also consider i is in register X_{22}.

```
A = X20
B = X21
i = X22
```

```
    Addi  X22, X0, 8
Loop:
    Beq   X22, X0, Exit

    If:
    Slli  X5, X22, 3      // offset calc.
    Add   X5, X5, X20     // Base + offset
    ld    X6, 0(X5)       // X6 = A[i]
    Bne   X6, X22, inc

    ld    X7, 24(X21)     // X7 = B[3]
    Slli  X7, X7, 3       // X7 = X7 x 2^3
    Add   X7, X7, X20     // Base + offset
    ld    X5, 0(X7)       // X5 = A[B[3]]
    sd    X5, 16(X20)     // A[2] = X5

inc:
    Addi  X22, X22, -1
    Beq   X0, X0, Loop

Exit:
```

Question - 3:

Construct the equivalent RISC-V code of the following C code.

```
if ( A[i] < i){
    A[2] = A [B[3]] ;
}
```

Base addresses of array A and B are in register X_{20} and X_{21} . Also consider i is in register X_{22}.

```
If :
slli  x5, x22, 3      // offset calc.
Add   x5, x5, x20     // Base + offset
ld    x6, 0(x5)       // x6 = A[i]
BGE   x6, x22, Exit

ld    x7, 24(x21)     // x7 = B[3]
slli  x7, x7, 3       // x7 = x7 x 2^3
Add   x7, x7, x20     // Base + offset
ld    x5, 0(x7)       // x5 = A[B[3]]
sd    x5, 16(x20)     // A[2] = x5

Exit :
```

A = X_{20}
B = X_{21}
i = X_{22}

Question - 4:

Construct the equivalent RISC-V code of the following C code.

```
if ( A[3] != A[6]){
    if (A[3] == 0) {
        A[3] = A[3] + 2;
    }else{
        A[6] = A[6] / 16;
    }
}else{
    A[6] = A[6] * 8
}
```

Base addresses of array A and B are in register X_{20}

If 1:

```
ld   x28, 24(x20)      // x28 = A[3]
ld   x29, 48(x20)      // x29 = A[6]
Beq  x28, x29, Else1
```

If 2:

```
ld   x5,  24(x20)      // x5 = A[3]
BNE  x5,  x0,  Else2
```

→ A[3] already loaded into x_5 and x_5 is unchanged till now.

```
Addi x5, x5, 2
Sd   x5, 24(x20)       // A[3] = x5 + 2
Beq  x0, x0, Exit
```

Else 2:

```
ld   x5, 48(x20)       // x5 = A[6]
SRLI x5, x5, 4         // x5 = x5 / 2^4
Sd   x5, 48(x20)       // A[6] = x5
Beq  x0, x0, Exit
```

Else 1:

```
ld   x28, 48(x20)      // x28 = A[6]
Slli x28, x28, 3       // x28 = x28 * 8
Sd   x28, 48(x20)      // A[6] = x28
```

Exit:

Question - 5:

Translate the following code written in C programming language into instructions sequence written in RISC-V Assembly. The values in the variables a, b, and c inside the add function are stored in the argument registers $\$X_{10}$, $\$X_{15}$, and $\$X_{12}$ respectively. Also, the values in the variables x and y inside the are stored in the argument registers $\$X_{13}$ and $\$X_{14}$ respectively. The return value is stored in the return register $\$X_{11}$ for both functions.

int max(int x, int y) {
  if(x > y)
  return x;
  else
  return y;
}
	int calc(int a, int b, int c) {
	  return a + max(b,c)
	}

x10 x15 x12

x13 x14

x13 < x14

Calc:

```
Addi   Sp, Sp, -8
sd     X1, 0(SP)
Addi   X13, X15, 0      [X13 = X15 + 0]   (b)
Addi   X14, X12, 0      [X14 = X12 + 0]   (c)
Jal    X1, max
Add    X11, X10, X11
ID     X1, 0(SP)
Addi   Sp, Sp, 8
Jalr   X0, 0(X1)
```

X10 = a; X11 = return from max

Max:

```
Blt    X13, X14, maxElse   (x, y)
Addi   X11, X13, 0
Beq    X0, X0, ExitMax

maxElse:
Addi   X11, X14, 0

ExitMax:
Jalr   X0, 0(X1)
```

Quer – 6

Write RISC-V assembly code that checks if the number stored in register $\mathbf{X_{25}}$ is **even** or not. If **even** then store **1** in register $\mathbf{X_{26}}$ otherwise store **0**.

0000 .. .. .. 0001 = 1

```
Addi  x27, x0, 1
And   x27, x25, x27 // LSB bit required
BNE   x27, x0, else
Addi  x26, x0, 1
Beq   x0, x0, Exit

else:
Addi  x26, x0, 0
Exit:
```

You can also use Slli / Srli to figure it out.

Quer - 7

ADD X_{25} , X_{25} , X_0 . Can you make this instruction faster? If yes, Write the updated instruction?

Yes, ADDI x_{25}, x_{25}, 0

0 is hard wired here.

Quer - 8

Memory Location	Code	Line Number	Machine Code
	ADDI $\mathbf{X_5}$, X_0 , 5	1	
# 7064	ADDI $\mathbf{X_6}$, X_0 , 1	2	
# 7068	ADDI $\mathbf{X_{25}}$, X_0 , 0	3	
# 7072	Loop: BLT $\mathbf{X_5}$, $\mathbf{X_6}$, loopBreak	4 -3×4 =-12	0 | 000 000 | 00110 | 00101 | XXX | 1000 | 0 | XXXXXXX (imm 12 | imm [10:5] | rs2 | rs1 | imm [4:1] | imm [1])
# 7076	ADDI X_{25} , X_{25} , 1	5	
#7080	ADDI X_5 , X_5 , -1	6	
	BEQ $\mathbf{X_0}$, $\mathbf{X_0}$, Loop	7	1 | 111 111 | 00000 | 00000 | XXX | 1010 | 1 | XXXXXXX (12 | 10:5 | 4:1 | 11)
	loopBreak:	8 +4×4 =16	

a) What is the value of **PC** while executing line2? Answer:
7064 [2]

b) Fill up the machine codes corresponding to line4 and line7 in the table above. [5]

16 = 0000 0001 0000
+16 = 0 0000 0001 0000 X
(12 | 11 | 10 | 5 | 4 | 1)

12 = 0000 0000 1100
+12 = 0 0000 0000 1100
= 1 1111 1111 0011
+ 1
-12 = 1 111111 1010 0 0x
(12 | 11 | 10 | 5 | 4 | 1)

Ques - 9

# Location		Line No.
	Loop :	
# 80000	Slli X_{10}, X_{22}, 3	1
# 80004	Add X_{10}, X_{10}, X_{25}	2
# 80008	Ld X_9, 0(X_{10})	3
# 80012	Bne X_9, X_{24}, Exit	4
# 80016	Addi X_{22}, X_{22}, 1	5
# 80020	Beq X_0, X_0, loop	6
	Exit:	
# 80024	⟶	7

rs1 rs2

SB type instructions are — Beq X_0, X_0, loop

Bne X_9, X_{24}, Exit

For Beq,

We are moving 5 instruction upward.

Hence, $-5 \times 4 = -20$

upward direction

20 = 0000 0001 0100

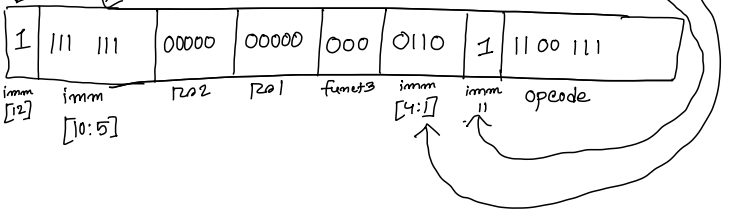
+20 = 0 0000 0001 0100

1 1111 1110 1011

+ 1

1 1111 1110 1100

12 11 10:5 4:1 0 x



For Bne,

We are moving 3 instructions downward

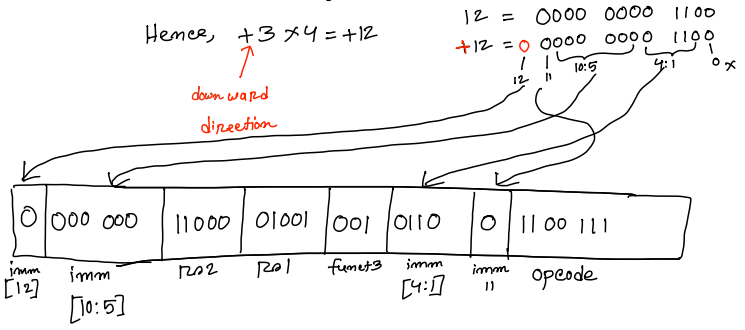
Hence, $+3 \times 4 = +12$

downward direction

12 = 0000 0000 1100

+12 = 0 0000 0000 1100

12 11 10:5 4:1 0 x



Quez -10

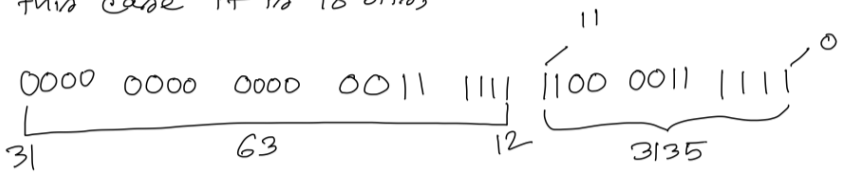
Write necessary RISC-V instructions to store the value $(1111\ 1111\ 0000\ 1111\ 11)_2$ in X20 register.

(18 bits)

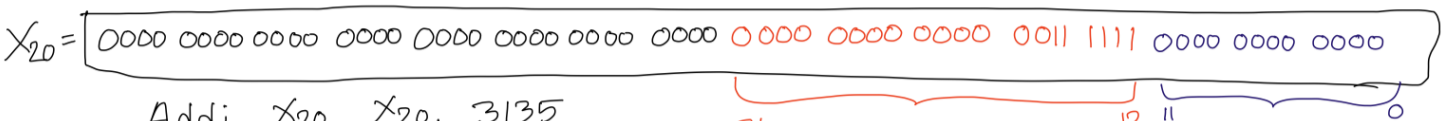
Answer2:

Check if the given number requires more than 12 bits to store.

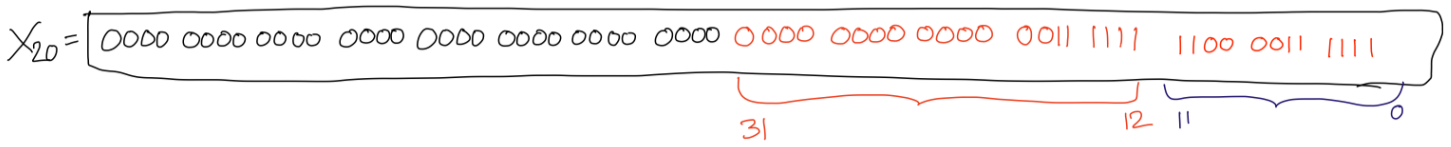
In this case it is 18 bits,



Lui X20, 63



Addi X20, X20, 3135



Quez-12

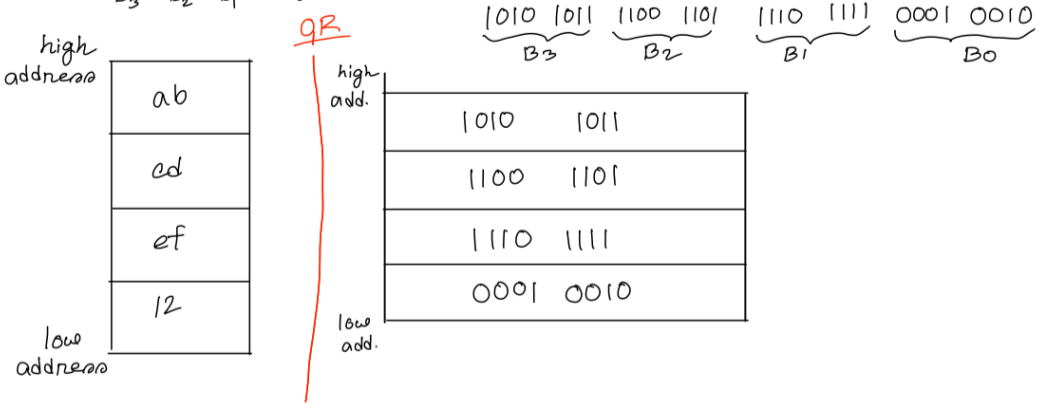
Show how the value 0xabcdef12 would be arranged in memory in RISC-V machine.

Answer3:

RISC-V follows Little endian approach to store data in memory

$\Rightarrow$ $(abcdef12)_{16}$

B_3 B_2 B_1 B_0



Ques - 12

For the RISC-V assembly instructions below, what is the corresponding C/high level statement?

slli x30, x5, 3 $\Rightarrow X_{30} = f \times 8$ add x30, x10, x30 $\Rightarrow X_{30} = Base_A + 8f$ slli x31, x6, 3 $\Rightarrow X_{31} = g \times 8$ add x31, x11, x31 $\Rightarrow X_{31} = Base_B + 8g$ ld x5, 0(x30) $\Rightarrow f = A[f]$ addi x12, x30, 8 $\Rightarrow X_{12} = X_{30} + 8$ ld x30, 0(x12) $\Rightarrow X_{30} = A[f+1]$ add x30, x30, x5 $\Rightarrow X_{30} = A[f+1] + A[f]$ sd x30, 0(x31) $\Rightarrow B[g] = A[f+1] + A[f]$	Assume that the variables f, g, h, i, and j are assigned to registers x5, x6, x7, x28, and x29, respectively. Assume that the base address of the Arrays A and B are in registers x10 and x11, respectively.

$B[g] = A[f+1] + A[f]$