CS2323 - Computer Architecture

Homework - 2

Q1

Ans a) addi x15, x22, -45

According to I-type format -

imm[11:0]	rs1	funct3	rd	opcode

∴ Opcode - $(0010011)_2$
rd - x15 - $(15)_{10}$ - $(01111)_2$
funct 3 - 0x0 - $(000)_2$
rs1 - x22 - $(22)_{10}$ - $(10110)_2$
imm - $(-45)_{10}$ - $(1111\ 1101\ 0011)_2$

∴ Machine code -

1111 1101 0011	10110	000	01111	0010011
imm[11:0]	rs1	f3	rd	opcode

= 0xfd3b0793

b) and x23, x8, x9

According to R format -

funct7	rs2	rs1	funct3	rd	opcode

2

∴ opcode - $(0110011)_2$
rd - x23 - $(23)_{10}$ - $(10111)_2$
funct3 - 0x7 - $(111)_2$
rs1 - x8 - $(8)_{10}$ - $(01000)_2$
rs2 - x9 - $(9)_{10}$ - $(01001)_2$
funct7 - 0x00 - $(0000000)_2$

∴ Machine code -

0000000	01001	01000	111	10111	0110011
f7	rs2	rs1	f3	rd	opcode

= 0x009847b3

c) blt x2, x11, 240

According to B format -

imm[12|10:5]	rs2	rs1	funct3	imm[4:1|11]	opcode

∴ opcode - $(1100011)_2$
imm - $(240)_{10}$ - $(00\ 000111\ 1000\ 0)_2$
funct3 - 0x4 - $(100)_2$
rs1 - x2 - $(2)_{10}$ - $(00001)_2$
rs2 - x11 - $(11)_{10}$ - $(01011)_2$

∴ Machine code -

imm[12]	imm[10:5]	rs2	rs1	f3	imm[4:1]	imm[11]
0	000111	01011	00001	100	1000	0

1100011
opcode

$= 0x0eb14863$

d) sd x19, -54(x1)

According to S format-

imm[11:5] rs2 rs1 funct3 imm[4:0] opcode

∴ opcode $\rightarrow (0100011)_2$
funct3 $\rightarrow 0x3 \rightarrow (011)_2$
rs1 $\rightarrow$ x1 $\rightarrow (1)_{10} \rightarrow (00001)_2$
rs2 $\rightarrow$ x19 $\rightarrow (19)_{10} \rightarrow (10011)_2$
imm $\rightarrow (-54)_{10} \rightarrow$ 1111110 01010

∴ Machine code -

imm[11:5]	rs2	rs1	f3	imm[4:0]
1111110	10011	00001	011	01010

0100011
opcode

$= 0xfd30b523$

4

e) jal x3, -10116

According to J format -

imm[20|10:1|11|19:12] rd opcode

∴ opcode - $(1101111)_2$
rd - x3 - $(3)_{10}$ - $(00011)_2$
imm - $(-10116)_{10}$ - 1 1111 1101 1
0000 11110 0

∴ Machine code

1 0000111110 1 11111101 00011
imm[20] imm[10:1] im[11] imm[19:12] rd
1101111
Opcode

= 0x87dfd1ef

Q2

Ans a) li x5, -1

→ addi x5, x0, -1
$(-1)_{10}$ is within the range of the 12-bit immediate ∴ It translates to a single addi instruction

b) li x5, 0xffffffff

→ addi x5, x0, -1
Immediate is signed → 0xffffffff is same as $(-1)_{10}$. Same as above (a)

c) li x5, 132

→ addi x5, x0, 132
$(132)_{10}$ is within range of the 12-bit immediate ∴ It translates to a single addi instruction.

d) li x5, 2134
→ ~~lu~~
→ lui x5 0x1
→ addiw x5, x5, -1962

$(2134)_{10}$ is not within the range of the 12 bit immediate ∴ We use

lui to set the upper 20 bits of a register (last 12 are set to 0)

∴ After loading 1 in x5
x5 = 0x00801000 i.e $(4096)_{10}$

Now, we subtract the excess i.e 1962 from x5

e) li x5, 0x2345abcd

→ lui x5, 0x2345b
→ addi x5, x5, -1075

As 0x2345abcd is not within the range of the 12bit immediate, we use lui to load 0x2345b (a greater value than the immediate) and subtract off the excess (i.e +1075)

Q3

Ans.

```
.data
.dword  0xa55aa5a593933939
.dword  0x39933939a55aa5a5

.text
lui x1, 0x10000

.main
lhu x3, 0(x1)
# loading half word from the first dword
with no offset. As it's unsigned the
extension is all 0's
x3 - 0x0000000000003939

lh x3, 0(x1)
# As the first digit is 3 ie (0011)2, the
signed extension of the loaded
half word is same as above.
x3 - 0x0000000000003939

lh x3, 2(x1)
# loading halfword from the first dword
with 2-byte offset. As it signed we
check the leading digit ie 9
(9)16 = (1001)2 → Sign extension will have
f's
x3 - 0xffffffffffff9393
```

ld x3, 0(x1)
# loading double word from first dword with no offset. As it's signed we check the leading digit i.e
No need to bother with extension as dword completely occupies the register.
x3 - a55a a5a5 9393 3939

lw x3, 12(x1)
# loading word from second dword with 4 byte offset. As its signed we check the leading digit i.e 3
$(3)_{16} = (0011)_2$ → Sign extension will be 0's
x3 - 0x00000000 39933939

lbu x3, 7(x1)
loading byte from first dword with 7-byte offset. As its unsigned the extension is all 0's

0x0000000000000a5

lb x3, 7(x1)
Same as above. As its signed we check leading digit i.e a
$(a)_{16} = (1010)_2$ → Sign extension will have f's

x3 - 0xffffffffffffff a5

lb x3, 6(x1)

loading byte from the first dword with 6-byte offset. As it signed we check the leading digit ie 5
$(5)_{10} = (0101)_2 \longrightarrow$ Sign extension is all 0's

x3 - 0x0000000000000005a

Q4-

Ans. No, it's not fixed (base address) and can be configured by going to Ripes > Edit > Settings > Compiler.
In the assembler section it allows you to set the base address for .text, .data and .bss section of the assembly code