CS2323: HW-2

VELMA DHATRI REDDY
AI21BTECH11030

1) (a) addi $x_{15}, x_{22}, -45$

$rd = x_{15}$ (01111) $\quad rs_1 = x_{22}$ (10110)

imm = -45 (111111010011)

111111010011	10110	000	01111	0010011
immediate	rs_1	func3	rd	opcode

= 0xFD3B0793

(b) and x_{23}, x_8, x_9

0000000	01000 1	01000	111	10111	0110011
func7	rs_2	rs_1	func3	rd	opcode

= ~~0x940BBB3~~ 0x00947BB3

(c) blt x_2, x_{11}, 240

~~1111000~~ imm = 12 11 10:5 4:1 0
0 0 000111 1000 0

~~imm~~

$rs_1 = x_2$ (00010) $\quad rs_2 = x_{11}$ (01011)

0 000111	01011	00010	100	1000 0	1100011
imm[12,10:5]	rs_2	rs_1	func3	imm[4:1,11]	opcode

= 0x0EB14863

(d) sd x_{19}, -54(x_1)

$rs_1 = x_1$ (00001) $\quad rs_2 = x_{19}$ (10011)

imm = -54 (1111110 01010)

1111110	10011	00001	011	01010	0100011
imm[11:5]	rs_2	rs_1	func3	imm[4:0]	opcode

~~(c)~~ = 0xFD30B523

(e) jal x_3, -10116

rd = x_3 (00011) imm = 1 1111 0000 1101 1 0000 1111 1100

imm = 1 11111101 1 0000111110 0

[20] [19:12] [11] [10:1] [0]

1 0000111110 1 11111101 00011 1101111

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

= 0x87DFD1EF

2(a) li x_5, -1

Disassembled code: addi x_5, x_0, -1

This is because -1 lies in the range of -2^{11} to $2^{11}-1$. It can be written as above as x_0 always has a value of 0

(b) li x_5, 0xFFFFFFFFF

Disassembled code: addi x_5, x_0, -1

0xFFFFFFFF is a 32 bit integer which is equal to -1. We need to load -1 to x_5 which same as the above.

(c) li x_5, 132

Disassembled code: addi x_5, x_0, 132

This is because 132 lies in the range of -2^{11} to $2^{11}-1$. It can be written as above as x_0 always has a value of 0

(d) li x_5, 2134

Disassembled code: lui x_5 0x1

addiw x_5, x_5, -1962

This is because 2134 doesn't lie in the range of -2^{11} to $2^{11}-1$. First instruction loads ~~0x1000~~ 0x100. lui appends 12 0s to the end. Value of x_5 is 4096 Addiw takes 12 bits immediate which is (-1962) and adds to x_5 which gives 2134.

e) li x5, 0x2345abcd

Disassembled code: lui x5, 0x23456

addiw x5, x5, -1075

This is similar to the previous one decimal representation of 0x23456 is greater than $2^{11}-1$. First instruction loads first 20 bits and appends 12 bits 0sat the end (i.e 0x23456000) addiw takes the 12 bit immediate and adds to x5 which gives 0x2345abcd

3) i) lhu x3, 0(x1)

x3 : 0x0000 0000 0000 3939

lhu loads, unsigned halfword (16 bits) ie into x3 : We load the first 16 bits which are 3939. Thus, x3 has a value of 0x0000 0000 0000 3939.

ii) lh x3, 0(x1)

x3: 0x0000 0000 0000 3939

lh loads signed half words into x3. First 16 bits are 3939.

$0x3939 \equiv (0011\ 1001\ 0011\ 1011)_2$

↓

0 (Hence, positive) (Rest of the bits are 0)

iii) lh x3, 2(x1)

x3 : 0x ffff ffff ffff 9393

lh loads signed half words into x3. Memory of x1 after offset of 2bytes is 0x9393

$0x9393 \equiv (1001\ 0011\ 1001\ 0011)_2$

as the first most significant is has 1 rest all are f.

iv) ld x_3, 0(x_1) x_3: 0x a55a a5a5 9393 3939

ld loads the doubleword from address at x_1, which gives the above value.

v) lw x_3, 12(x_1) x_3: 0x 0000 0000 3993 3939

An offset of 12 leads us to 32 bits of second double word which is (3993 3939) hence, the above value

vi) lbu x_3, 7(x_1) x_3: 0x 0000 0000 0000 00a5

lbu loads 1byte after an offset of 7 bytes from address at x_1 (from the first double words) hence, the above value.

vii) lb x_3, 7(x_1) x_3: 0x ffff ffff ffff ffa5

lb loads 1byte (signed) after offset of 7 bytes from address at x_1, which 0xa5 = 0b10100101 (as it is starting with 1 rest all are f's) Hence, the above value

viii) lb x_3, 6(x_1) x_3: 0x 0000 0000 0000 005a

lb loads 1byte (signed) after an offset of 6bytes from address at x_1 which is 0x5a = 0b 01011010 (as the most significant is 0 rest all are 0's) Hence, the above value.

4) NO, it is not fixed . It is configurable by the ~~bel~~ below steps

Edit → settings → compiler → .data start address