

CS2323: Computer Architecture, Autumn 2024

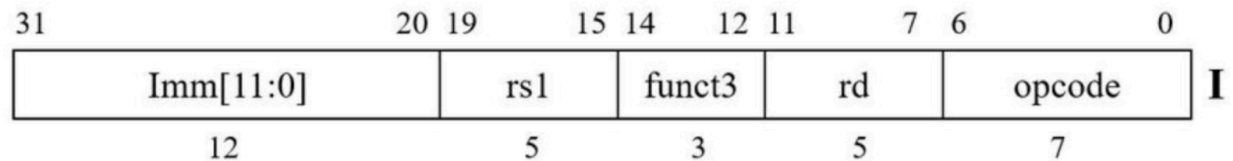
Homework-3: RISC-V Assembly Encoding

Ahmik Virani
ES22BTECH11001

Q1)

(a) addi x15, x22, -45

This is I format



Immediate is -45 i.e. we have to find 2's complement of 45,

45 in binary is = 0000 0010 1101

1's complement = 1111 1101 0010

2's complement = 1111 1101 0011

-45 will contribute the immediate complement as 1111 0010 0011

rs1 register is x22, 22 in binary is 10110

rs1 part will be 10110

funct3 for addi is 0x0 so will be 000

rd register is x15, 15 in binary is 1111

rd part will be 01111

opcode for addi is 0010011

Finally combining all

Immediate	rs1	funct1	rd	opcode
111111010011	10110	000	01111	0010011

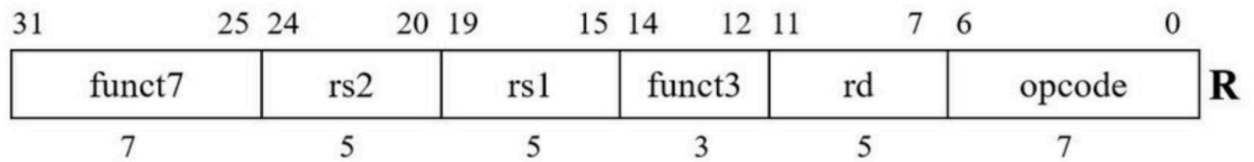
To make into hex, lets first combine them 4 at a time

1111 1101 0011 1011 0000 0111 1001 0011

In hexa decimal : 0xFD3B0793

(b) and x23, x8, x9

and is R format



funct7 is 0x00, which in binary is 00000000

rs2 is x9, in binary 01001

rs1 is x8, in binary 01000

funct3 for and is 0x7, in binary 111

rd is x23, in binary 10111

opcode for and is 0110011

funct7	rs2	rs1	funct3	rd	opcode
00000000	01001	01000	111	10111	0110011

Combining: 0000 0000 1001 0100 0111 1011 1011 0011

In hexadecimal: 0x00947BB3

(c) blt x2, x11, 240

blt is B format

imm[12 10:5]	rs2	rs1	funct3	imm[4:1 11]	opcode	B-type
--------------	-----	-----	--------	-------------	--------	--------

Immediate is 240, in binary is 0000011110000

rs2 is x11, in binary is 01011

rs1 is x2, in binary is 00010

funct3 for blt is 0x4, in binary is 100

Opcode for blt is 1100011

imm[12 10:5]	rs2	rs1	funct3	imm[4:1 11]	opcode
0000111	01011	00010	100	10000	1100011

Combining: 0000 1110 1011 0001 0100 1000 0110 0011

In hexadecimal: 0x0EB14863

(d) sd x19, -54(x1)

It is S format

imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode
7 bits	5 bits	5 bits	3 bits	5 bits	7 bits

Immediate is -54,
 54 is 0000 0011 0110 in binary
 1's complement 1111 1100 1001
 2's complement 1111 1100 1010
 So -54 is 1111 1100 1010 in binary

rs1 is x1, in binary 00001
 rs2 is x19, in binary 10011
 funct3 for sd is 0x3, in binary is 011
 opcode for sd is 0100011

imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode
1111110	10011	00001	011	01010	0100011

Combining it: 1111 1101 0011 0000 1011 0101 0010 0011
 In hexadecimal: 0xFD30B523

(e) jal x3, -10116
 jal is J format

imm[20:10:11:19:12]	rd	opcode	J-type
---------------------	----	--------	--------

Immediate is -10116
 First find 10116 in binary (Each line we keep dividing by 2)

10116 % 2 = 0
 5058 % 2 = 0
 2529 % 2 = 1
 1264 % 2 = 0
 632 % 2 = 0
 316 % 2 = 0
 158 % 2 = 0
 79 % 2 = 1
 39 % 2 = 1
 19 % 2 = 1
 9 % 2 = 1
 4 % 2 = 0

$2 \% 2 = 0$

1

So in binary: 10011110000100

But immediate contributes 21 bits,

So rewriting it: 0 0000 0010 0111 1000 0100

1's complement: 1 1111 1101 1000 0111 1011

2's complement: 1 1111 1101 1000 0111 1100

Finally -10116 is 1 1111 1101 1000 0111 1100 in binary

rd is x3, in binary 00011


opcode for jal is 1101111

imm[20 10:1 11 19:12]	rd	opcode
10000111110111111101	00011	1101111

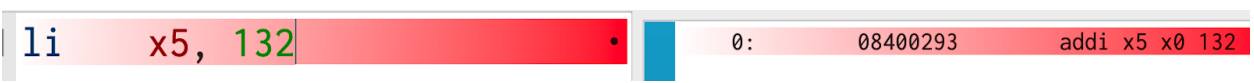
Combining: 1000 0111 1101 1111 1101 0001 1110 1111

In hexadecimal: 0x87DFD1EF

Q2)

(a) 

The immediate -1, is small and it fits into the 12 bits signed, so addi alone will suffice. So basically what is happening is we want to load -1 into x5, so we are just adding the value of $-1 + 0$, resulting to the destination register of x5

(b) 

Again a very similar explanation as part (a), 132 in binary is 0000 1000 0100, and clearly fits into the 12 bits signed value that addi register can accommodate.

(c) 

Now here the scenario changes as we 2134 in binary is 1000 0101 0110, which will have a different meaning in signed. Addi immediate value is signed, but here we require it to be unsigned, so we must use lui, which is load upper immediate, which loads the bits shifted by 12. Now clearly, the most significant bit is 1, which for addi will mean that the

value is negative, so to ensure that the final value is positive, we need to load 2^{12} , which is loading upper immediate of 1, to ensure that negative bias is removed.

1000 0101 0110 signed is $-1 * (2^{11}) + 2^6 + 2^4 + 2^2 + 2^1 = -1962$

So that initial 1 that we loaded by shifting it on the 12th bit allows us to accurately ensure that $2^{12} - 1962 = 2134$, which is the required amount we need to load

(d)

Here we are loading a 32 bit value.

0x2345abcd in binary is: 0010 0011 0100 0101 1010 1011 1100 1101

We know that addi can accommodate only the lower 12 bits that too signed.

Addi can accommodate 1011 1100 1101. But again the same problem as before. We need to get rid of the bias, we need to load the upper bits incremented by 1.

Therefore we will do an lui of 0010 0011 0100 0101 1011 instead of 0010 0011 0100 0101 1010.

0010 0011 0100 0101 1011 in hexadecimal is: 0x2345b

And addi will add 1011 1100 1101 as signed, which is -1075 in decimal.

Q3)

(a) 0x0019F233

Converting to binary: 0000 0000 0001 1001 1111 0010 0011 0011

Least significant 7 bits for Opcode: 0110011 -> R format

funct7	rs2	rs1	funct3	rd	opcode	R-type
--------	-----	-----	--------	----	--------	--------

Next 5 bits for rd: 00100, i.e. x4

Next 3 bits for funct3: 111, i.e. 0x7 -> and instruction

Next 5 bits rs1: 10011, i.e. x19

Next 5 bits rs2: 00001, i.e. x1

Next 7 bits funct7: 0000 000, i.e. 0x00

Instruction: and rd, rs1, rs2

i.e. and x4, x19, x1

(b) 0x06B4D763

Converting to binary: 0000 0110 1011 0100 1101 0111 0110 0011

Least significant 7 bits for Opcode: 1100011 -> B format

imm[12 10:5]	rs2	rs1	funct3	imm[4:1 11]	opcode	B-type
--------------	-----	-----	--------	-------------	--------	--------

imm[4:1|11] is next 5 bits: 01110

funct3 is next 3 bits: 101, i.e. 0x5 -> bge instruction

rs1 is next 5 bits: 01001, i.e. x9

rs2 is next 5 bits: 01011, i.e. x11

imm[12|10:5] is next 7 bits: 0000011

Now arranging the immediate bits: 0000001101110 (The last bit is assumed to be zero as immediate values are even numbers)

Immediate value: $2^1 + 2^2 + 2^3 + 2^5 + 2^6 = 110$

Instruction bge rs1, rs2, imm

I.e bge x9, x11, 110

(c) 0x0169CF93

Converting to binary: 0000 0001 0110 1001 1100 1111 1001 0011

Least significant 7 bits for Opcode: 0010011 -> I format

imm[11:0]	rs1	funct3	rd	opcode	I-type
-----------	-----	--------	----	--------	--------

Next 5 bits for rd: 11111, i.e. x31

Next 3 bits for funct3: 100, i.e. 0x4 -> xori instruction

Next 5 bits for rs1, i.e. 10011, i.e. x19

Next 12 bits for immediate value: 0000 0001 0110, which in decimal equals $2^1 + 2^2 + 2^4 = 22$

Instruction: xori rd, rd1, imm

I.e. xori x31, x19, 22