NYU Tandon School of Engineering

Fall 2021, ECE 6913

Homework Assignment 2

Instructor: Azeez Bhavnagarwala, email: ajb20@nyu.edu

Course Assistant Office Hour Schedule (Room 808, 370 Jay St: 9AM – 11AM)

Mondays & Tuesdays: Haotian (Kenny) Zheng **hz2687@nyu.edu** & Shan Hao **sh6206@nyu.edu,**

Wednesdays: Karan Parikh **kap9580@nyu.edu**

Thursday: Sahil Chitnis **ssc9983@nyu.edu**

Fridays: Kewal Jani **kj2062@nyu.edu**

Saturday: Zhiming Fan **zf2035@nyu.edu**

[released Friday September 17th 2021] [due* **Sunday September 26th 2021, *before* 11:55 PM**]

You *are allowed* to discuss HW assignments only with other colleagues taking the class. You are *not allowed* to share your solutions with other colleagues in the class. Please feel free to reach out to the Instructor during office hours or by appointment if you need any help with the HW.

Please enter your responses in this Word document after you download it from NYU Classes. *Please use the NYU Classes portal to upload your completed HW. Please do not upload images of handwritten sheets or PDFs of scanned sheets of handwritten solutions. Please be sure to type-in your solutions into Word or Google Docs and upload machine readable documents only.*

- *In RISCV, only load and store instructions access memory locations*
- *These instructions must follow a 'format' to access memory*
- *Assume a 32-bit machine in all problems unless asked to assume otherwise*

Problem 1:

Assume address in memory of 'A[0]', 'B[0]' and 'C[0]') are stored in Registers x27, x30, x31. Assume values of variables f, g, h, i,and jare assigned to registers x5, x6, x7, x28, x29 respectively

Write down RISC V Instruction(s) to

(a).Load Register x5*with content* of A[10]

ld x5,40(x9)

(b).Store contents of Register x5into A[17]

sw x5,68(x9)

(c).add 2 operands: one in x5 - a register, the other in in Register x6. Assume result of operation to be stored in register x7

add x7,x5,x6

(d).copy contents at one memory location to another: C[g] = A[i+j+31]

Add x28,x28,x29 #i+j
Addi x28,x28,31 #i+j+31
Slli x28,x28,2 #4*(i+j+31)
Add x28,x28,x27 #x28 has &A[i+j+31]
Slli x6,x6,2 #g=g*4g

```
Add x6,x6,x31        #x6 has &C[g]
Lw x28,0(x28)        #load contents of &A[i+j+31]
Sw x28,0(x26)        #write contents into C[g]
```

(e). implement in RISC V these line of code in C:

(i) f = g - A[B[9]]

```
Lw x8,36(x30)     #read contents of B[9]
Slli x8,x8,2
Add x8,x27,x8     #x8 has &A[B[9]]
Lw x8,0(x30)      #read contents
Sub x5,x6,x8
```

(ii) f = g - A[C[8] + B[4]]

```
Lw x30,16(x30)
Lw x31,32(x31)
Add x24,x30,x31
Slli x24,x24,2
Add x4,x27,x24
Lw x4,0(x4)
Sub x5,x6,x4
```

(iii) A[i] = B[2i+1], C[i] = B[2i]

```
Add x26,x28,x28   #2i
Addi x26,x26,1    #2i+1
Slli x26,x26,2
Add x26,x30,x26   #&B[2i+1]
Slli x25,x28,2
Add x25,x27,x25   #&A[i]
Lw x24,0(x26)     #B[2i+1]
Sw x24,0(x25)     #A[i]=B[2i+1]

Addi x26,x26,-4   #&B[2i]
Slli x23,x28,2
Add x23,x23,x31  #&C[i]
Lw x24,0(x26)    #B[2i]
Sw x24,0(x23)    #C[i]=B[2i]
```

(iv) A[i] = 4B[i-1] + 4C[i+1]

```
Addi x26,x28,-1  #i-1
Slli x26,x26,2
Add x26,x30,x26 #&B[i-1]
Addi x25,x25,1   #i+1
Slli x25,x25,1
Add x25,x25,x31 #&C[i+1]
Add x24,x24,x27 #&A[i]
Lw x23,0(x26)    #B[i-1]
Slli x23,x23,2
Lw x22,0(x25)   #C[i+1]
Slli x22,x22,2
Add x23,x23,x22 #4B[i-1]+4C[i+1]
Sw x23,0(x24)
```

(v) f = g - A[C[4] + B[12]]

```
Lw x30,48(x30)  #B[12]
Lw x31,16(x31)  #C[4]
Add x24,x30,x31
Slli x24,x24,2
Add x4,x27,x24
```

Lw x4,0(x4)
Sub x5,x6,x4

Problem 2:

Assume the following register contents:

`x5 = 0x00000000AAAAAAAA, x6 = 0x1234567812345678`

a For the register values shown above, what is the value of `x7` for the following sequence of instructions?

```
srli x7, x5, 16
```

X7 0x000000000000AAAA

```
addi x7, x7, -128
```

X7 (0000101010101000101010)2

```
srai x7, x7, 2
```

x7 (00000010101010001010)2

```
and x7, x7, x6
```

x7 = (10 0000 1000)2

b For the register values shown above, what is the value of `x7` for the following sequence of instructions?

```
slli x7, x6, 4
```

x7 = 0x2345678123456780

c For the register values shown above, what is the value of `x7` for the following sequence of instructions?

```
srli x7, x5, 3
```

X7 = (1010101010101)2

```
andi x7, x7, 0xFEF
```

X7 = (010101000101)2

Problem 3:

For each RISC-V instruction below, identify the instruction format and show, wherever applicable, the value of the opcode (op), source register (rs1), source register (rs2), destination register (rd), immediate (imm), func3, func7 fields. Also provide the 8 hex char (or 32 bit) instruction for each of the instructions below

add x5, x6, x7
addi x8, x5, 512
ld x3, 128(x27)

sd x3, 256(x28)
beq x5, x6 ELSE #ELSE is the label of an instruction 16 bytes larger
#than the current content of PC
add x3, x0, x0
auipc x3, FFEFA
jal x3 ELSE

instruction	Type	Opcode,func3,func7	Rs1	Rs2	rd	imme
1	r	0x33,0x0,0x0	6	7	5	-
2	i	0x13,0x0,-	5	-	8	512
3	i	0x3,0x3,-	27	-	3	128
4	s	0x23,0x3,-	28	-	3	256
5	sb	0x63,,0,-	5	6	-	16
6	r	0x33,0x0,0x0	0	0	3	-
7	u	0x17,-,-	-	-	3	0XFFEFA
8	uj	0x6f,-,-	-	-	3	16

8 hex characters of above 8 instructions:

1.0x007302B3

2. 0x20028413

3. 0x080DB183

4. 0x103E3023

5. 0x00628863

6. 0x000001B3

7. 0xFFEFA197

8. 0x010001EF

Problem 4:

(a) For the following C statement, write a minimal sequence of RISC-V assembly instructions that performs the identical operation. Assume `x5 = A`, and `x11` is the base address of C.

`A = C[0] << 16;`

//X11 &C[0]

Lw x5,0(x11)

Slli x5,x5,16

(b) Find the shortest sequence of RISC-V instructions that extracts `bits 12 down to 7` from register `x3` and uses the value of this field to replace bits `28 down to 23` in register `x4` without changing the other bits of registers `x3` or `x4`. (Be sure to test your code using `x3 = 0` and `x4 = 0xffffffffffffffff`. Doing so may reveal a common oversight.)
Add x7,x0,0x3f
Slli x7,x7,7

And x28,x3,x7
Slli x7,x7,16
Xori x7,x7,-1
And x4,x4,x7 //23-28 turn to zero
Slli x28,x28,16
Or x4,x28,x4

(c) Provide a minimal set of RISC-V instructions that may be used to implement the following pseudoinstruction:

```
not x5, x6 // bit-wise invert
```

xor x5,x6,-1

[Hint: note that there is no 'not' instruction in RISCV. However, an XOR immediate instruction could be used]

Problem 5:

Suppose the program counter (PC) is set to $\texttt{0x60000000}_{\texttt{hex}}$.

a. What range of addresses can be reached using the RISC-V *jump-and-link* (`jal`) instruction? (In other words, what is the set of possible values for the PC after the jump instruction executes?)

0X5FF00000-0x600FFFFE

The immediate number in jal instruction stored as imm[20:1], and the imm[0] are always set as default value 0.

So the maximum positive number of immediate number is:

0 1111 1111 1111 1111 1110 = 0x0FFFFE,

The minimum negative number of immediate number is:

1 0000 0000 0000 0000 0000

0x60000000 XOR 0xFFF00000

0x60000000 XOR 0x000FFFFE

b. What range of addresses can be reached using the RISC-V *branch if equal* (`beq`) instruction? (In other words, what is the set of possible values for the PC after the branch instruction executes?)

The maximum positive number of immediate number is:

0 1111 1111 1110 = 0x0FFE

The minimum negative number of immediate number is:

1 0000 0000 0000

0x5FFFF000~0x60000FFE

Problem 6:

Assume that the register `x6` is initialized to the value 10. What is the final value in register x5 assuming the `x5` is initially zero?

```
LOOP:     beq x6, x0, DONE
          addi x6, x6, -1
          addi x5, x5, 2
          jal x0, LOOP
DONE:
```

The final value in register x5 is 20

a. For the loop above, write the equivalent C code. Assume that the registers `x5` and `x6` are integers `acc` and `i`, respectively.

```
Int acc=0;
For (int i=10;i>0;i--){
  Acc+=2
}
```

b. For the loop written in RISC-V assembly above, assume that the register `x6` is initialized to the value N. How many RISC-V instructions are executed?

After exiting from the loop, one more instruction corresponding to the DONE label is executed.Thus the anwser 4N+1

c. For the loop written in RISC-V assembly above, replace the instruction "`beq x6, x0, DONE`" with the instruction "`blt x6, x0, DONE`" and write the equivalent C code.

```
LOOP: blt x6, x0, DONE
      addi x6, x6, -1
      addi x5, x5, 2
      jal x0, LOOP
DONE:
```

```
Int acc=0;
For (int i=10;i>=0;i--){
  Acc+=2
}
```

Problem 7:

a Translate the following C code to RISC-V assembly code. Use a minimum number of instructions. Assume that the values of `a, b, i,` and `j` are in registers `x5, x6, x7,` and `x29,` respectively. Also, assume that register `x10` holds the base address of the array `D`.

```
for(i=0; i<a; i++)
    for(j=0; j<b; j++)
        D[4*j] = i + j;
```

```
LOOPI:
 addi x7, x0, 0 // Init i = 0
 bge x7, x5, ENDI // While i < a
 addi x30, x10, 0 // x30 = &D
 addi x29, x0, 0 // Init j = 0
LOOPJ:
 bge x29, x6, ENDJ // While j < b
 add x31, x7, x29 // x31 = i+j
 sd x31, 0(x30) // D[4*j] = x31
```

```
addi x30, x30, 32 // x30 = &D[4*(j+1)]
addi x29, x29, 1 // j++
jal x0, LOOPJ
ENDJ:
addi x7, x7, 1 // i++;
jal x0, LOOPI
ENDI:
```

b How many RISC-V instructions does it take to implement the C code from 7a. above? If the variables **a** and **b** are initialized to **10** and **1** and all elements of **D** are initially 0, what is the total number of RISC-V instructions executed to complete the loop?

The code require 13 RISIC-V instructions.

When a is 10 and b is 1,the total instructions are 123. 12*10+3

Problem 8:

Consider the following code:

```
lb x6, 0(x7)
sd x6, 8(x7)
```

Assume that the register x7 contains the address **0×10000000** and the data at address is 0×**11**22334455667788.

a What value is stored in 0×10000007 on a bigendian machine?
0x88

b What value is stored in **0×10000007** on a littleendian machine?
0x11

Problem 9:

Write the RISC-V assembly code that creates the 64-bit constant $0x1234567812345678_{hex}$ and stores that value to register x10.

Lui x10,0x12345

Addi x10,x10,0x678

Slli x10,x10,32

Lui x5,0x12345

Addi x5,x5,0x678

Add x10,x10,x5

Problem 10: Assume that **x5** holds the value $\mathbf{128_{10}}$.
a. For the instruction **add x30, x5, x6,** what is the range(s) of values for **x6** that would result in overflow?
Assume a 32-bit machine, x5 =128
x6 should be a positive number to make overflow happens.
X6 = (2^32)-1-128 = (2^32)-129

If x6>(2^32)-129, there is an overflow

b. For the instruction **sub x30, x5, x6,** what is the range(s) of values for **x6** that would result in overflow?
If x6<-(2^32)+129, there is an overflow
c. For the instruction **sub x30, x6, x5,** what is the range(s) of values for **x6** that would result in overflow?

If x6<-(2^32)+128, there is an overflow