NYU Tandon School of Engineering

Spring 2021, ECE 6913 – Section B

Homework Assignment 2

Course Assistant: Archit, Deep: ax368@nyu.edu dss569@nyu.edu

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

[released Monday February 8th 2021] [due* **Wednesday February 17th 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 `j` are 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]`

(b) Store contents of Register `x5` into `A[17]`

(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`

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

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

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

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

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

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

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

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
addi x7, x7, -128
srai x7, x7, 2
and x7, x7, x6
```

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

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

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

```
srli x7, x5, 3
andi x7, x7, 0xFEF
```

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
```

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;
```

(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.)

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

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

[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 $0x60000000_{hex}$.

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?)

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?)

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:
```

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

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?

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.

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;
```

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?

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**22334455667**88**.

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

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

Problem 9:

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

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?

b. For the instruction **`sub x30, x5, x6,`** what is the range(s) of values for **x6** that would result in overflow?

c. For the instruction **`sub x30, x6, x5,`** what is the range(s) of values for **x6** that would result in overflow?

SOLUTIONS

1.

a. lw x5, 40(x27)

b. sw x5, 68(x27)

c. add x7, x5, x6

d.

add x28, x28, x29

addi x28, x28, 31

slli x28, x28, 2

add x28, x28, x27

slli x6, x6, 2

add x6, x6, x31

lw x28, 0(x28)

sw x28, 0(x6)

e. 1.

lw x5, 36(x30)

slli x5, x5, 2

add x5, x5, x30

lw x5, 0(x5)

sub x5, x6, x5

e.2.

lw x30, 16(x30)

lw x31, 32(x31)

add x24, x30, x31

slli x24, x24,2

```
add x27, x27, x24
lw x27, 0(x27)
sub x5, x6, 27
```

e.3.

```
add x26, x28, x28
addi x26, x26, 1
slli x26, x26,2
add x26, x26, x30
slli x25, x28, 2
add x25, x25, x27
lw x24, 0(x26)
sw x24, 0(x25)
addi x26, x26, -4
slli x23, x28, 2
add x23, x23, x31
lw x24, 0(x26)
sw x24, 0(x23)
```

e.4.

```
#A[0]', 'B[0]' and 'C[0]') x27, x30, x31.
#f, g, h, i, and j x5, x6, x7, x28, x29

addi x26, x28, -1
slli x26, x28, 2
add x26, x26, x30
lw x26, 0(x26)
```

slli x26, x26, 2

addi x25, x28, 1

slli x25, x28, 2

add x25, x25, x31

lw x25, 0(x25)

slli x25, x25, 2

add x26, x26, x25

add x28, x28, x27

sw x28, 0(x26)

e.5.

lw x30, 48(x30)

lw x31, 16(x31)

add x24, x30, x31

slli x24, x24,2

add x27, x27, x24

lw x27, 0(x27)

sub x5, x6, x27

2. a

srli x7, x5, 16 ----->000000000000AAAA

addi x7, x7, -128 ----->000000000000AA2A

srai x7, x7, 2 ----->0000000000002AAA

and x7, x7, x6 ----->0000000000000228

2. b

0000000AAAAAAAA0

2. c

0000000999999990

AND

0000000000000FEF

=

0000000000000980

3.

	Op	Rs1	Rs2	Rd	Imm	Func3	Func7	Hex
Add	0110011	00110	00111	00011	-	000	0000000	33/0/00
addi	0010011	00101	-	01000	001000000000	000	-	13/0
Ld	0000011	11011	-	00011	000010000000	011	-	03/3
Sd	0100011	11100		00011	000100000000	011	-	23/3
Beq	1100011			00101	-	000	-	63/0
Add	0110011			00011	-	000	0000000	33/0/00
Auipc	0010111			00011	-	-	-	17
jal	1101111			00011	-	-	-	6F

4.

a. slli x5, x11, 16

b.

andi x10, x3, 00001F80

andi x11, x4, E0EFFFFF

or x4, x10, x11

c.

xori x5, x6, FFFFFFFF

5. -2^{18} to$2^{18} - 1$ or 600FFFFE to 5FF00001

b. 60000FFE to 5FFFF001

6.

x5 will have 20

a. for(i=10; i<=0; --i)
acc=acc+2;

b. 4n+1

c. for(i=10; i<0; --i)
acc=acc+2;

7.

LOOPi:

addi x7, x0, 0

bge x7, x5, ENDi

addi x30, x10, 0

addi x29, x0, 0

LOOPj:

bge x29, x6, ENDj

add x31, x7, x29 sd x31, 0(x30)

addi x30, x30, 32

addi x29, x29, 1

jal x0, LOOPj

ENDj:

addi x7, x7, 1

jal x0, LOOPi

ENDi:

b. It needed 13 RISC V Instructions. 123 instructions executed in the given case.

8.

a. Loading the first byte on Big-endian will get ‘11’ in x6. Sd instruction puts the value in x6 in 0x10000008. The value stored in 0x10000007 will be **0x88**

b. Loading the first byte on Little-endian will get ‘88’ in x6. Sd instruction puts the value in x6 in 0x10000008. The value stored in 0x10000007 will be **0x11**

9.

```
lui x10, 0x12345
addi x10, x10, 0x678
slli x10, x10, 16
slli x10, x10, 16
lui x5, 0x12345
addi x5, x5, 0x678
add x10, x10, x5
```

10. a. FFFFFFFF to FFFFFF80

b. -FFFFFFFF to -FFFFFF80

c. -FFFFFFFF to -FFFFFF7F