NYU TANDON SCHOOL OF ENGINEERING

Review Problems for Quiz 1

RISC V Instruction Sets

Azeez Bhavnagarwala

ECE 6913, Fall 2023

Computer Systems Architecture

NYU Tandon School of Engineering

Example 1

Instruction(s) to **load Register** x5 *with content* of Array data structure ‘A[6]’ stored *in memory*? Assume base address in memory of Array data structure ‘A’ (or address in memory of of ‘A[0]’) is stored in Register x9

```
lw x5, 24(x9)
```

Instruction(s) to **store contents of Register** x5 *into array element* A[1].Assume base address in memory of Array data structure ‘A’ (or address in memory of ‘A[0]’) is stored in Register x9

```
sw x5, 4(x9)
```

Example 2

Instruction(s) to **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
```

Instruction(s) to **subtract one operand from another**: *one specified in the 12 bit immediate field of the instruction, the other in memory location* ‘A[240]’ Assume base address in memory of Array data structure ‘A’ (or address in memory of ‘A[0]’) is stored in Register x28

```
lw x5, 960(x28)
```

```
addi x5, x5, -imm
```

Example 3

Instruction(s) to **copy contents at one memory location to another:**

`B[g] = A[i+j-3]`

Assume `i, j, g` values are in registers `x5, x6, x7`. Assume base address in memory of Array data structures '`A, B`' (or address in memory of '`A[0]`' and '`B[0]`') are stored in Registers `x28, x29`

```
add x5, x5, x6      # i+j
addi x5, x5, -3     # i+j-3
slli x5, x5, 2      # 4*(i+j-3)
add x5, x5, x28
slli x7, x7, 2      # g = 4*g
add x7, x7, x29
lw x5, 0(x5)
sw x5, 0(x7)
```

Example 4

Instruction(s) to implement code in C such as

```
f = g - A[B[4]];
```

Assume `f, g` values are in registers `x5, x6`. Assume base address in memory of Array data structures '`A, B`' (or address in memory of '`A[0]`' and '`B[0]`') are stored in Registers `x28, x29`

```
lw x7, 16(x29)
slli x7, x7, 2
add x7, x7, x28
lw x7, 0(x7)
sub x5, x6, x7
```

Example 5

In the next 5 problems, assume that the variables `f, g, h, i,` and `j` are assigned to registers `x5, x6, x7, x28, x29` respectively. Assume base address in memory of Array data structures '`A, B`' (or address in memory of '`A[0]`', '`B[0]`' and '`C[0]`') are stored in Registers `x27, x30, x31.` Write RISCV code that implements:

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

`A[i] = 2B[i-1] + 4C[i+1]`

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

`A[i] = B[4i+2], C[i] = B[4i]`

`A[i] = B[i-1] + C[i+1]`

```
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, x26, x30    #B[2i+1]
slli x25, x28, 2
add x25, x25, x27    #A[i]
lw x24, 0(x26)
sw x24, 0(x25) #A[i]=B[2i+1]
-----------------------------------------
addi x26, x26, -4    8i
add x26, x26, x30    #B[2i]
slli x23, x28, 2
add x23, x23, x31    #C[i]
lw x24, 0(x26)
sw x24, 0(x23) #C[i]=B[2i]
```

```
A[i] = 2B[i-1] + 4C[i+1]
addi x26, x28, -1  # i-1
slli x26, x26, 2
add x26, x26, x30  #B[i-1]
addi x25, x28, 1   # i+1
slli x25, x25, 2
add x25, x25, x31  #C[i+1]
slli x24, x28, 2
add x24, x24, x27  #A[i]
lw x23, 0(x26)     # B[i-1]
slli x23, x23, 1   #2B[i-1]
lw x22, 0(x25)     # C[i+1]
slli x22, x22, 2   #4C[i+1]
add x23, x23, x22  # 2B+4C
sw x23, 0(x24)     # A
```

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


lw x30, 16(x30)   #B[4]
lw x31, 32(x31)   #C[8]
add x24, x30, x31
slli x24, x24,2
add x4, x27, x24 #A
lw x4, 0(x4)
sub x5, x6, x4
```

- A[i] = B[4i+2], C[i] = B[4i]

```
slli x26, x28, 2        #4i
addi x25, x26, 2        #4i+2
slli x25, x25, 2        #4(4i+2)
add x3, x30, x25        #&B[4i+2]
lw x25, 0(x3)           #B[4i+2]
add x4, x27, x26        #A[i]
sw x25, 0(x4)           #A
-------------------------------
addi x3, x3, -8         #&B[4i]
lw x25, 0(x3)           #B[4i]
add x31, x31, x26       #&C[i]
sw x25, 0(x31)          #C[i] = B[4i]
```

```
A[i] = B[i-1] + C[i+1]
addi x18, x28, -1      #B[i-1]
slli x18, x18,2
add x18, x18, x30
lw x18, 0(x18)
addi x19, x28, 1       #C[i+1]
slli x19, x19, 2
add x19, x19, x31
lw x19, 0(x19)
add x18, x18, x19
slli x20, x28, 2
add x20, x20, x27
sw x18, 0(x20)         #A[i] = B[i-1] + C[i+1]
```

Example 6

Show how the value **0xedcba321** would be arranged in memory of a `little-endian` and a `big-endian` machine. Assume the data are stored starting at address 0 and that the *word size is 4 bytes*.

Little Endian:

B3	B2	B1	B0
ed	cb	a3	21

Big Endian:

B3	B2	B1	B0
21	a3	cb	ed

Example 7

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?

```
slli x7, x5, 4
or x7, x7, x6
```

x7 = 0000000A AAAAAAA0

(A hex character has 4 bits, shift left of 4 pushes the hex string left by one character while filling in the vacancy created on the right with a '**0**')

```
or x7, x7, x6
```

0000000A AAAAAAA0 OR 12345678 12345678

Yields in x7:

1234567A BABEFEF8

Example 7

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

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

Again, shift left by 4 bits moves the hex string to the left by one character, while inserting a ‘0’ into the vacancy on the right

using `x6 = 0x1234567812345678`, the above `slli` instruction produces:

```
23456781 23456780
```

Example 7

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
```

using x5 = 0x00000000AAAAAAAA
above srli instruction produces in the last 4 bytes:
initially,
x5 = 10101010 10101010 10101010 10101010
after right shift instruction:
x7 = 0001 0101 0101 0101 0101 0101 0101 0101
= 15555555
0xFEF = 0000 0000 0000 0000 0000 1111 1110 1111
andi x7, x7, 0xFEF
yields
x7 = 0000 0000 0000 0000 0000 0101 0100 0101
= 0x545

Example 8

For each RISC-V instruction below show the value of the `opcode (op)`, source register (`rs1`), and destination register (`rd`) fields in the 32 bit machine instruction. For the I-type instructions, show the value of the immediate field, and for the R-type instructions, show the value of the second source register (rs2). For non U and UJ-type instructions, show the funct3 field, and for R-type and S-type instructions, also show the funct7 field

```
addi x30, x10, 8 // x30 = &A[1]
op:0010011
rs1:01010
rd:11110
func3:000
immediate field:000000001000
```

Example 9

Find the shortest sequence of RISC-V instructions that extracts bits 16 down to 11 from register x5 and uses the value of this field to replace bits 31 down to 26 in register x6 without changing the other bits of registers x5 or x6. (Be sure to test your code using x5 = 0 and x6 = 0xffffffffffffffff. Doing so may reveal a common oversight.)

```
addi x7, x0, 0x3f       // Create bit mask for bits 16 to 11
slli x7, x7, 11         // Shift the masked bits
and x28, x5, x7         // Apply the mask to x5
slli x7, x7, 15         // Shift mask to cover bits 31 to 26
xori x7, x7, -1         // This is a NOT operation
and x6, x6, x7          // "Zero out" positions 31 to 26 of x6
slli x28, x28, 15       // Move selection from x5 into
                        // positions 31 to 26
or x6, x6, x28          // Load bits 31 to 26 from x28
```