Q1 RISC-V Warmup

5 Points

REMINDER: Gradescope does not have submission history for in-browser assignments such as lecture questions and homeworks. If you press "save answer" or "submit" after the deadline, the autograder *will* mark it as late and you'll get 0 points in the next TCP rerun. When you review old homework and lecture assignments, do not click any buttons otherwise you risk getting a zero on that assignment.

Answer the following questions about RISC-V. Don't be afraid to take some RISCs.

RISC-V is a(n) _ language.

- [x] machine code
- [ ] high-level
- [x] assembly
- [ ] C-like
- [ ] numerical
- [ ] None of the above

RISC-V follows the _____ CPU design strategy.

- [ ] CISC
- [x] RISC
- [ ] SISC
- [ ] DISC
- [ ] None of the above

Which of the following is true of the RISC philosophy? Select all that apply.

- [x] RISC stands for Reduced Instruction Set Computing
- [ ] RISC encourages the instruction set to be large and expansive
- [x] In RISC, a simple instruction set helps to build faster hardware
- [x] In RISC, complicated operations can be performed by composing simple instructions
- [ ] None of the above

Which of the following is False about a processor register?

- [x] A register is a small location of RAM memory.
- [x] A register has access speed comparable to a hard drive.
- [x] Registers comprise most of a computer's storage space.
- [x] A computer typically has thousands of registers.
- [ ] A register is a small amount of storage directly on the CPU.
- [ ] None of the above

RISC-V has __#__ integer registers.

32

For the RISC-V we learn in this class, a register can hold __#__ bytes.

4

Q2 Instruction Types

9 Points

Classify each of the following RISC-V instructions by type.

addi t0, a0, 1

- ○ R
- ◉ I
- ○ S
- ○ SB
- ○ U
- ○ UJ

---

lw t0, 8(s2)

- ○ R
- ◉ I
- ○ S
- ○ SB
- ○ U
- ○ UJ

---

beq a0, a1, loop

- ○ R
- ○ I
- ○ S
- ◉ SB
- ○ U
- ○ UJ

---

sb t0, 12(s2)

- ○ R
- ○ I
- ◉ S
- ○ SB
- ○ U
- ○ UJ

---

jal x0, label

- ○ R
- ○ I
- ○ S
- ○ SB
- ○ U
- ◉ UJ

---

bne x10,x0,loop

- ○ R
- ○ I
- ○ S
- ◉ SB
- ○ U
- ○ UJ

---

jalr ra, s0, 0

- ○ R
- ◉ I
- ○ S
- ○ SB
- ○ U
- ○ UJ

---

slli x11, x10, 4

- ○ R
- ◉ I
- ○ S
- ○ SB
- ○ U
- ○ UJ

---

sub t0, t1, zero

- (●) R
- ( ) I
- ( ) S
- ( ) SB
- ( ) U
- ( ) UJ

Q3 Misc Risc

4 Points

What does the following do:

```
xor x1, x1, x2
xor x2, x1, x2
xor x1, x1, x2
```

- [ ] Sets x1 to 0
- [ ] Sets x1 to -1
- [ ] Sets x2 to 0
- [ ] Sets x2 to -1
- [x] Swaps x1 and x2
- [ ] Sets `x1 == x2`
- [ ] Sets x1 and x2 to only bits which both do not share
- [ ] Sets x1 and x2 to only bits which both share

Given the following code:

```
# x10 holds 0x34FF
slli   x12, x10, 0x10
srli   x12, x12, 0x08
and    x12, x12, x10
```

What is in `x12`?

- ( ) 0x0
- (●) 0x3400
- ( ) 0x4F0
- ( ) 0xFF00
- ( ) 0x34FF
- ( ) None of the other options

Given the following code sequence (assume `x5` is somewhere on the stack):

```
addi x11, x0, -1261
sw x11, 0(x5)
sw x11, 2(x5)
```

```
lb x12, 1(x5)
lbu x13, 2(x5)
```

What is the 32bit value in `x12`? (Write your answer in hex)

0xFFFFFFFB

What is the 32bit value in `x13`? (Write your answer in hex)

0x00000013

Q4 Take a RISC for a WHILE

3 Points

C code:

```
1. int A[20];
2. int sum = 0;
3. int i = 19;
4. while ( i >= 0)  {
5.     sum += A[i];
6.     i --;
7. }
```

Corresponds to the following RISC-V assembly code:

```
1.  add x9, x0, x0
2.  add x10, x0, 19
3.  add x11, x0, x0
4.  Loop:
5.      bxx x10, x0, Done
6.      lw x12, 0(x11)
7.      addi x10, x10, value1
8.      addi x11, x11, value2
9.      add x9, x9, x12
10.    j Loop
11. Done:
```

Q4.1 For the while loop to operate correctly, the following needs to be true.

3 Points

`bxx` (on line 5) is

- ○ bgt
- ◉ blt
- ○ bleu
- ○ bgtu
- ○ bltu
- ○ beq
- ○ bne
- ○ bge
- ○ bgeu

`value1` (on line 7) is

- ○ -4
- ○ -2
- ◉ -1
- ○ 1
- ○ 2
- ○ 4
- ○ 16

`value2` (on line 8) is

- ○ -4
- ○ -2
- ○ -1
- ○ 1
- ○ 2
- ◉ 4
- ○ 16

Q5 RISC-V → Machine Code

5 Points

Convert each of the following RISC-V instructions to machine code.

Format your answer as 32-bit hexadecimal number, e.g. 0xABADCAFE or 0xabadcafe (the cafe just sucks). Leave any unused bits as 0.

Refer to your RISC-V Green Sheet if needed.

Q5.1

1 Point

Convert:

```
sub s0, t0, t2
```

0x40728433

Q5.2

1 Point

Convert:

```
slli a0, t0, 3
```

0x00329513

Q5.3

1 Point

Convert:

```
sw t0, 4(s0)
```

0x00542223

Q5.4
1 Point

Convert:

```
jal ra, fib
```

Assume $\mathrm{PC} = 0\mathrm{x}00400014$, and fib is located at $0\mathrm{x}00400028$.

0x014000EF

Q5.5
1 Point

Convert:

```
beq t3, t5, NEXT
```

Assume $\mathrm{PC} = 0\mathrm{x}0040011C$, and $NEXT$ is located at $0\mathrm{x}0040010C$.

0xFFEE08E3

Q6 Machine Code → RISC-V Instructions
4 Points

Convert each of the following 32-bit machine code instructions into a RISC-V instruction.

Formatting: use register names in your final answer, e.g. a0 instead of x10. The only exception is the zero register, which you should refer to as "x0" instead of "zero".

Example Format: "add t0, a0, x0" or "lw x0, 61(x5)"

For all questions, you can assume

$PC = 0\mathrm{x}00400010$

and we have the following labels:

$func1: 0\mathrm{x}00400000$

$func2: 0\mathrm{x}00400018$

NOTE: Until Gradescope updates its checking, please make sure to follow the formatting above.

Q6.1
1 Point

`0x00598523`

sb t0, 10(s3)

Q6.2
1 Point

`0x00b542b3`

xor t0, a0, a1

Q6.3
1 Point

`0x00728463`

beq t0, t2, func2

Q6.4
1 Point

`0xFF1FF06F`

jal x0, func1

Q7 C ↔ RISC-V
4 Points

In each of the following questions, you'll be given either C or RISC-V code, and asked to examine a translation into RISC-V or C, respectively.

The translation may have a bug or logic error. Your task is to identify the line number containing the error, or enter "none" if no problem exists.

Given this C code:

```
if (x < 5) {
    goto label;
}
```

Identify the buggy line in the following RISC-V translation. Select "None", if there is no error.

```
/* x = s0 */
1. addi t0 x0 5
2. slt t1 s0 t0
3. bne t1 x0 label
```

- [ ] Line 1
- [ ] Line 2
- [ ] Line 3
- [x] None

Given this C code:

```
int x[2] = {1, 2}
int y = x[0] + x[1] + 2;
```

identify the buggy line in the following RISC-V translation or enter "none", if there is none:

```
/* x = s0, y = s1 */
1. lw t0 0(s0)
2. lw t1 1(s0)
3. add s1,t0,t1
4. addi s1,s1,2
```

- [ ] Line 1
- [x] Line 2
- [ ] Line 3
- [ ] Line 4
- [ ] None

Given this RISC-V code:

```
sub s1 s2 s1
add s0 s0 s2
j end
```

identify the buggy line in the following C translation or enter "none", if there is none. (Hint: register order matters):

```
/* x = s0, y = s1, z = s2 */
1. y -= z;
2. x += z;
3. goto end;
```

- [x] Line 1
- [ ] Line 2
- [ ] Line 3
- [ ] None

Given this RISC-V code:

```
slli s0 s0 4
xori s0 s0 3
sw s0 12(s1)
```

identify the buggy line in the following C translation or enter "none", if there is none:

```
1. int x; // x = s0
2. int y[5]; // y = s1
3. x *= math.pow(2, 4);
4. x ^= 3;
5. y[4] = x;
```

- [ ] Line 1
- [ ] Line 2
- [ ] Line 3
- [ ] Line 4
- [x] Line 5
- [ ] None

Q8 RISC-V Green Sheet questions

6 Points

Which of the following is the correct ordering of fields for R-type instruction from bits 31 to 0?

- [x] funct7, rs2, rs1, funct3, rd, opcode
- [ ] opcode, rs2, rs1, funct3, rd, funct7
- [ ] funct7, rd, rs1, funct3, rs2, opcode
- [ ] funct7, rs1, rs2, funct3, rd, opcode
- [ ] None of the above

Which of the following RISC-V instructions is actually a pseudo-instruction?

- [x] li
- [ ] mulh
- [ ] addi
- [ ] ori
- [ ] None of the above

Which of the following fields is NOT used in I-Type instructions?

- [ ] rs1
- [x] rs2
- [ ] rd
- [ ] opcode
- [ ] None of the above

Which of the following registers are preserved across a call? (i.e. should have the same value before and after you make a RISC-V function call)

- [ ] a0
- [x] s0
- [ ] t4
- [x] x19
- [x] x9
- [ ] x31
- [ ] None of the above

Which of the following RISC-V instructions includes a conditional statement in its operation?

- [x] beq
- [x] slt
- [ ] lui
- [x] sltu
- [ ] None of the above

Which of the following ordering is correct for memory allocation starting from 0x0?

- [ ] Text/Code, Static Data, Stack, Heap
- [ ] Static Data, Text/Code, Stack, Heap
- [x] Text/Code, Static Data, Heap, Stack
- [ ] None of the above

HW04 - RISC-V

GRADED

STUDENT
Sofia Anuar

TOTAL POINTS
40 / 40 pts

QUESTION 1
RISC-V Warmup — **5** / 5 pts

QUESTION 2
Instruction Types — **9** / 9 pts

QUESTION 3

Misc Risc **4** / 4 pts

QUESTION 4

Take a RISC for a WHILE **3** / 3 pts

4.1 For the while loop to operate correctly, the following needs to be true. **3** / 3 pts

QUESTION 5

RISC-V → Machine Code **5** / 5 pts

5.1 (no title) **1** / 1 pt

5.2 (no title) **1** / 1 pt

5.3 (no title) **1** / 1 pt

5.4 (no title) **1** / 1 pt

5.5 (no title) **1** / 1 pt

QUESTION 6

Machine Code → RISC-V Instructions **4** / 4 pts

6.1 (no title) **1** / 1 pt

6.2 (no title) **1** / 1 pt

6.3 (no title) **1** / 1 pt

6.4 (no title) **1** / 1 pt

QUESTION 7

C ↔ RISC-V **4** / 4 pts

QUESTION 8

RISC-V Green Sheet questions **6** / 6 pts