LAB 5

By Gerardo Garcia de Leon

ENCM 369

Lab Section B01

EXERCISE A:

1. The machine code for the instruction is:

imm11:0	rs1	funct3	rd	op
1111_1010_0000	_0001_0	000	_0001_0	001_0011

This structure for the instruction was found on page 335 of the textbook. The imm value is equal to -96 in binary. The rs1 and the rd represent sp which is found to be register x2 using Table B.4. The opcode and the funct3 was found on page 335.

2. The inputs are:

0111_1111_1111_1111_1110_1110_0100_0000 (0x7fff_ee40)

1111_1111_1111_1111_1111_1111_1010_0000 (-96)

—---

(1) 0111_1111_1111_1111_1110_1101_1110_0000 (0x7fff_ede0)

This answer was obtained by converting the hexadecimal into its respective binary representation, and then performing a two’s complement on 96 to obtain its negative value. These were then summed up and we obtained the new result for the address of the sp.

EXERCISE B:

Part 1:

a = 1110_0010

b = 1110_0001

Result:

(1) 1100_0011

- This has no signed overflow, but does have unsigned overflow

There is no unsigned overflow as the result has the same MSB value as the two operands, but there is unsigned overflow as the result is smaller than the operands, which wouldn't make logical sense

Part 2:

a = 0010_0000

b = 0101_0000

Result:

(0) 0111_0000

- This has no signed or unsigned overflow

There is no signed overflow as the operands and the result both have the same MSB and there is no unsigned overflow as the result is larger than both operands and has a carryout of 0

Part 3:

a = 0111_0110

b = 0000_1010

Result:

(0) 1000_0000

- There is signed overflow, but no unsigned overflow

There is signed overflow because both operands are positive but the result is negative, but there is no unsigned overflow as the result is larger than both operands and has a carryout of 0

Part 4:

a = 1101_0010

b = 1010_0101

Result:

(1) 0111_0111

- There is both signed and unsigned overflow

There is signed overflow as the result has the opposite MSB (operands are negative, result is positive) and there is unsigned overflow as the result is smaller than both operands and the carryout is 1

EXERCISE E:

FOR EACH PART, -b WAS OBTAINED BY DOING TWO'S COMPLEMENT

Part 1:

a = 1001_0000

b = 0001_0101 → -b = 1110_1011

Result:

(1) 0111_1011

- There is signed overflow, but no unsigned overflow.

There is signed overflow as the operands have a different MSB than the result. There is unsigned overflow as the result is smaller than the original operands and has a carryout of 1

Part 2:

a = 0100_0000

b = 1011_0000 → -b = 0101_0000

Result:

(0) 1001_0000

- There is signed overflow, but no unsigned overflow

There is signed overflow as the result is negative when both of our operands are positive, but there is no unsigned overflow as the result is larger than both of our operands, with a carryout of zero

Part 3:

a = 1101_0000

b = 1111_0000 → -b = 0001_0000

Result:

(0) 1110_0000

- There is no signed or unsigned overflow

There is no signed overflow as the we are adding a small positive number (16) to a larger negative number (-48) and should result in another negative number, which we correctly get (-32) and there is no unsigned overflow as the result is larger than both operands and the carryout is zero

Part 4:

a = 1111_0100

b = 1111_0010 → -b = 0000_1110

Result:

(1) 0000_0010

- There is no signed overflow, but there is unsigned overflow

There is no signed overflow as we are adding 14 to -12 and should obtain a positive number (2) which we did. There is unsigned overflow as the operand a is larger than the result and our carryout of 1

EXERCISE G

```
# void int2str(char *dest, int str)
#    arg/var                  GPR
#      dest                    a0
#          src                 a1
#      unsigned abs_src t0
#          unsigned ten     t1
#          unsigned rem    t2
#          unsigned temp             t3
#          char *p             t4
#      char *q                 t5
#
# Remark: We have used up all but one of the t-registers.
# However, a2-a7 can also be used for intermediate results.
#
        .text
        .globl   int2str
int2str:
        bne      a1, zero, L2
        li       t6, 48
        sb       t6, (a0)
        sb    zero, 1(a0)
```

```
        j       L10

L2:
        li      t6, 0x80000000
        bne     a1, t6, L3
        sub     t0, zero, t6
        j       L5
L3:
        bge     a1, zero, L4
        sub     t6, zero, a1
        mv      t0, t6
        j       L5
L4:
        mv      t0, a1
L5:
        mv      t4, a0
        li      t1, 10
L6:
        beq     t0, zero, L7
        remu    t2, t0, t1
        la      a2, digits
        add     a2, a2, t2
        lb      a3, (a2)
        sb      a3, (t4)
        addi    t4, t4, 1
        divu    t0, t0, t1
        j       L6
L7:
        bge     a1, zero, L8
```

```
        li      a4, 0x2d
        sb      a4, (t4)
        addi    t4, t4, 1
L8:
        li      a4, 0
        sb      a4, (t4)
        addi    t5, t4, -1
        mv      t4, a0
L9:
        bge     t4, t5, L10
        lb      t3, (t4)
        lb      a5, (t5)
        sb      a5, (t4)
        mv      a6, t3
        sb      a6, (t5)
        addi    t4, t4, 1
        addi    t5, t5, -1
        j       L9
L10:
        jr      ra
```

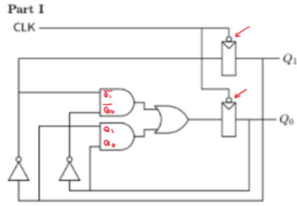
EXERCISE H:

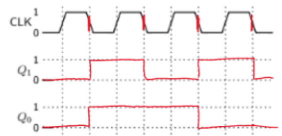
SEND

July 28, 2024 2:57 PM

Worksheet for Exercise H

Part I

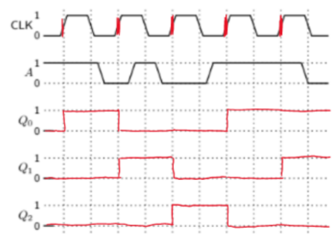


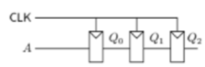


$Q_1' = \overline{Q_1}$

$Q_0' = (Q_1 \cdot Q_0) + (\overline{Q_1} \cdot \overline{Q_0})$

Part II





$Q_0' = A$

$Q_1' = Q_0$

$Q_2' = Q_1$