Jose Franco Baquera

February 28, 2018

CS - 273

Homework #5

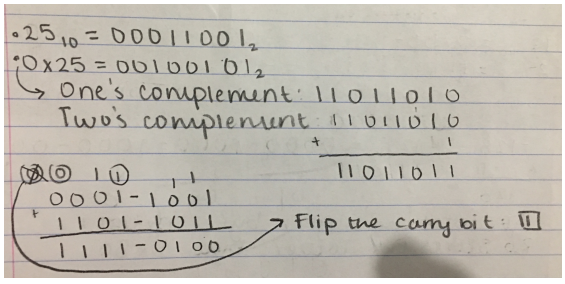
1. **What is the consequence if a computer architecture does not have a status register like the one in AVR?**

If a computer architecture does not have a status register like the one in the AVR, then it is impossible for it do to conditional branching. The implications of not having conditional branching would be extremely disadvantageous, especially since it would not allow the computer architecture to execute non-linear programs. In addition, without a status register, a computer architecture cannot detect overflow in arithmetic calculations, meaning that it would be incapable of making accurate decisions every time it is given an input. This implies that any form of non-human controlled devices (e.g. artificial intelligence, auto-pilot navigation, etc.) would be impossible to operate correctly without a status register.

2. **Determine the C, Z, V, S, N flags of the status after the following instructions:**

a.
```
LDI R18, 25
CPI R18, 0x25
```

We note that 0x25 is a hexadecimal number and its equivalent decimal number is equal to 37. To find the values of the flags in the status register, we must basically "subtract" (i.e. using two's complement) the value 37 from 25. NOTE: We must flip the carry bit at the end of the subtraction.



Answer:

C = 1 (Carry flag. Flip the bit since we are "subtracting.")

Z = 0 (Zero flag)

V = 0 (Overflow flag)

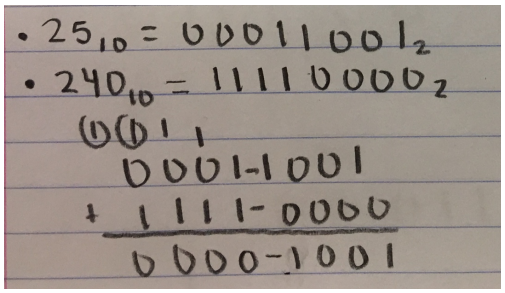
S = 1 (Sign flag)

N = 1 (Negative flag)

(NOTE: The H flag was not asked for this assignment, but it would be equal to 1.)

b.
```
LDI R18, 25
LDI R19, 240
ADD R18, R19
```

For this problem, we must only add the two decimal values 25 and 240 (using binary math again) and look at the specific flags.



Answer:

C = 1 (Carry flag)

Z = 0 (Zero flag)

V = 0 (Overflow flag)

S = 0 (Sign flag)

N = 0 (Negative flag)

(NOTE: The H flag was not asked for this assignment, but it would be equal to 0.)

3. **Determine the value of R1 and R0 after the following multiplications**

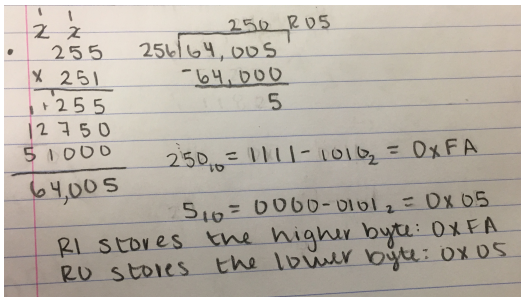
a.
```
LDI R20, -1
LDI R21 -5,
MUL R20, R21
```

It is important to note that the MUL assembly instruction will treat the numbers as unsigned. The bit patter for -1 is 1111-1111, which is equivalent to 0xFF, or 255 in decimal. The bit pattern for -5 is 1111-1011, which is equivalent to 0xFB, or 251 in decimal.

Answer:

R1 = 0xFA

R0 = 0x05



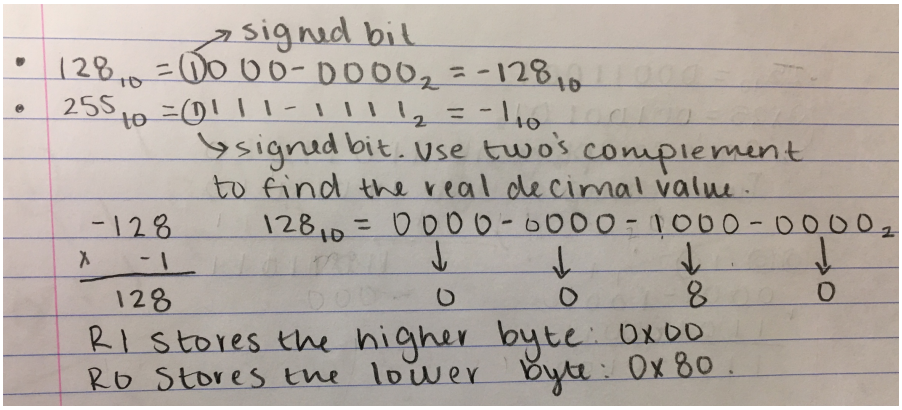
b.
```
LDI R20, 128
LDI R21, 255
MULS R20, R21
```

It is important to note that the MULS assembly instruction will treat the numbers as signed. The bit patter for 128 is 1000-0000, but since the assembly instruction treats this as a signed number, it is equivalent (by using two's complement) to -128. The bit pattern for 255 is 1111-1111, but since the assembly instruction treats this as a signed number, it is equivalent (by using its two's complement) to -1.

Answer:

R1 = 0x00

R0 = 0x80



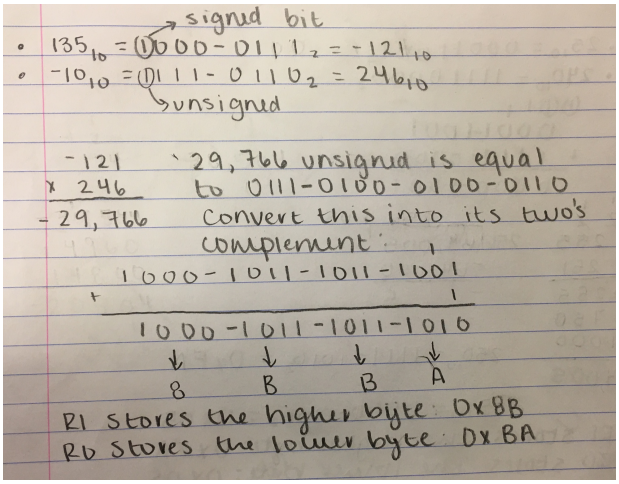
c.
```
LDI R20, 135
LDI R21, -10
MULSU R20, R21
```

It is important to note that the MULSU assembly instruction will treat the first operand as a signed number while the second operand as an unsigned number. The bit pattern for 135 is 1000-0111, but since the assembly instruction treats this as a signed number, it is equivalent (by using two's complement) to -121 in decimal. The bit pattern for -10 is 1111-0110, but since the assembly instruction treats this as unsigned, it is equivalent to 246 in decimal.

Answer:

R1 = 0x8B

R0 = 0xBA



4. **Revise the unsigned integer division in class and write a program to take two signed numbers A / B and return the quotient and the remainder.**

Here is the assembly program:

```
LDI R20, 112
LDI R21, -9

SUBI R20, 0
BRMI numeratorNegative ;Numerator is negative.
JMP numeratorPositive ;Numerator is positive.

numeratorNegative:
  SUBI R21, 0
  BRMI numNegDenNeg ;Both numerator and denominator are negative.
  JMP numNegDenPos ;Numerator is negative, denominator is positive.

numeratorPositive:
  SUBI R21, 0
  BRMI numPosDenNeg ;Numerator is positive, denominator is negative.
  JMP numPosDenPos ;Both numerator and denominator are positive.

numNegDenNeg:
  NEG R20
  NEG R21
  CLR R22

nnDivision:
  INC R22
  SUB R20, R21
  BRCC nnDivision
  DEC R22
  ADD R20, R21
  STS quotient,R22
  STS remainder,R20
  RET

numPosDenNeg:
  NEG R21
  NEG R20
  CLR R22

pnDivision:
  DEC R22
  ADD R20, R21
  BRCC pnDivision
```

```
  INC r22
  SUB R20, R21
  STS quotient,R22
  NEG R20
  STS remainder,R20
  RET

numNegDenPos:
  CLR R22

npDivision:
  DEC R22
  ADD R20, R21
  BRCC npDivision
  INC R22
  SUB R20, R21
  STS quotient, R22
  STS remainder, R20
  RET

numPosDenPos:
  CLR R22

ppDivision:
  INC R22
  SUB R20, R21
  BRCC ppDivision
  DEC R22
  ADD R20, R21
  STS quotient,R22
  STS remainder,R20
  RET
```

(NOTE: R22 will have the quotient while R20 will have the remainder.)

5. **Convert the following C code to assembly. Both x and y must be treated as signed numbers.**

```
int x=5, y=-5;
if(x > y) {
    y = x + 2;
} else if (x > y - 3) {
    y = x + 5
```

```
    } else {
        y = x + 7
    }
```

Here is the equivalent assembly code:

```
    LDI R20, 5 ;x
    LDI R21, -5 ;y

    CP R21, R20
    BRLT ifBody
    JMP elseIf

ifBody:
    MOV R21, R20
    LDI R22, 2
    ADD R21, R22
    JMP end

elseIf:
    MOV R22, R21
    SUBI R22, 3
    CP R22, R20
    BRLT elseIfBody
    JMP elseBody

elseIfBody:
    MOV R21, R20
    LDI R22, 5
    ADD R21, R22
    JMP end

elseBody:
    MOV R21, R20
    LDI R22, 7
    ADD R21, R22

end:
    JMP end
```