ECE 375 LAB 5

Large Number Arithmetic

Lab Time: Thursday 1000-1200

Eric Prather

PRELAB QUESTIONS

1. *For this lab, you will be asked to perform arithmetic operations on numbers that are larger than 8 bits. To be successful at this, you will need to understand and utilize many of the various arithmetic operations supported by the AVR 8-bit instruction set. List and describe all of the addition, subtraction, and multiplication instructions (i.e. ADC, SUBI, FMUL, etc.) available in AVR's 8-bit instruction set.*

- ADC: Add with carry
- ADD: Add without carry
- ADIW: Add immediate to word
- INC: Increment
- FMUL: Fractional multiply unsigned
- FMULS: Fractional multiply signed
- FMULSU: Fractional multiply signed with unsigned
- MUL: Multiply Unsigned
- MULS: Multiply signed
- LSL: Multiply by 2 (logical shift left)
- MULSU: Multiply signed with unsigned
- SBC: Subtract with carry
- SBCI: Subtract immediate with carry
- SBIW: Subtract immediate from word
- SUB: Subtract without carry
- SUBI: Subtract immediate

Fractional multiplication uses any radix.

2. Write pseudocode for an 8-bit AVR function that will take two 16-bit numbers (from data memory addresses $0111:$0110 and $0121:$0120), add them together, and then store the 16-bit result (in data memory addresses $0101:$0100). (Note: The syntax "$0111:$0110" is meant to specify that the function will expect little-endian data, where the highest byte of a multi-byte value is stored in the highest address of its range of addresses.)

Here is the pseudocode:

```
ADD16_PREDEFINED_ADDRESSES: ;uses r0, r1, Z, and mpr
        Z = $0110
        R0 = *(Z++)
        R1 = *(Z)
        Z = $0120
        mpr = *(Z++)
        r31 = *(Z--) ; saving space
        r30 = mpr ; for clarity, since we aren't using z to read anymore
        add r1 r31 // We start with the least significant (highest) byte.
        adc r0 r30 // Also adds in the previous carry bit.
        Z = $0100
        *(Z++) = r0
        *(Z++) = r1
```

3. Write pseudocode for an 8-bit AVR function that will take the 16-bit number in $0111:$0110, subtract it from the 16-bit number in $0121:$0120, and then store the 16-bit result into $0101:$0100.

```
SUB16_PREDEFINED_ADDRESSES: ;uses r0, r1, Z, and mpr
        Z = $0120
        R0 = *(Z++)
```

```
R1 = *(Z)
Z = $0110
mpr = *(Z++)
r31 = *(Z--) ; saving space
r30 = mpr ; for clarity, since we aren't using z to read anymore
sub r1 r31 // We start with the least significant (highest) byte.
subc r0 r30 // Also adds in the previous carry bit.
Z = $0100
*(Z++) = r0
*(Z++) = r1
```