ECE 375 PRELAB 5

Lab Time: Wednesday 12-2

Bradley Martin

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 adds two registers and the contents of the C flag.

   ADD- Add without carry only adds two registers.

   ADIW-Add immediate to word will add a value to a register pair and places the result in the register pair.

   DEC-Decrement subtracts one from the register given.

   FMUL-Fractional Multiply Unsigned does 8 bit x 8 bit unsigned multiplication.

   FMULUS-Fractional Multiply Signed does 8bit x 8 bit signed multiplication.

   INC-Increment adds one to the register.

   LSL-Logical Shift Left shift the bits one to the left multiplying by 2.

   MUL- Multiply Unsigned performs 8 bit x 8 bit unsigned multiplication.

   MULS- Multiply Signed performs 8 bit x 8 bit Signed multiplication.

   MULSU- Multiply Signed with Unsigned performs 8 bit x 8 bit multiplication of a signed and unsigned number.

   ROL-Rotate Left trough Carry shifts all bits to the left multiplying multi-byte signed and unsigned values by two

   SBC-Subtract with Carry subtracts two registers with the C Flag.

   SBCI-Subtract Immediate with carry SBI subtracts a constant from a register and subtracts with the C flag.

   SBIW- Subtract immediate from word subtracts a value from a pair of registers and places the result in the register pair.

   SUB-Subtract without carry only subtracts two registers.

   SUBI-Subtract Immediate subtracts a register and a constant and places the result in the destination register.

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.)

-Load 0x0111 into a register

-load 0x0110 into a register

-load 0x0121 into a register

-load 0x0120 into a register

- add the registers with 0x0111 with 0x0121 and add 0x110 with 0x0120

-store the registers of 0x0111 and 0x0110 into 0x0101 and 0x0100

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.

-Load 0x0111 into a register

-load 0x0110 into a register

-load 0x0121 into a register

-load 0x0120 into a register

- subtract the registers with 0x0111 with 0x0121 and add 0x110 with 0x0120

-store the registers of 0x0111 and 0x0110 into 0x0101 and 0x0100

REFERENCE