ECE 375 PRELAB 4

Lab Time: Tuesday 4-6

Alexander Uong

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.

Addition: ADD, ADC, ADIW

ADD: Add without Carry

Adds two registers without the C Flag, placing the result in the destination register Rd.

Rd← Rd + Rr

ADC: Add with Carry

Adds two registers and the contents of the C Flag, placing the result in the destination register Rd.

Rd←Rd+Rr+C

ADIW: Add Immediate to Word

Adds an immediate value (0 - 63) to a register pair and places the result in the register pair. This instruction operates on the upper four register pairs.

Rd+1:Rd←Rd+1:Rd+K

Subtraction: SUB, SUBI, SBC, SBCI, SBIW

SUB: Subtract Without Carry

Subtracts two registers, placing the result in the destination register Rd

Rd ← Rd – Rr

SUBI: Subtract Immediate

Subtracts a register and a constant, placing the result in the destination register Rd. This instruction is working on Register R16 to R31.

Rd ← Rd - K

SBC: Subtract With Carry

Subtracts two registers and subtracts with the C Flag, placing the result in the destination register Rd.

Rd ← Rd - Rr – C

SBCI: Subtract Immediate with Carry SBI – Set Bit in I/O Register

Subtracts a constant from a register and subtracts with the C Flag, placing the result in the destination register Rd.

Rd ← Rd - K – C

SBIW: Subtract Immediate from Word

Subtracts an immediate value (0-63) from a register pair, placing the result in the register pair. This instruction operates on the upper four register pairs.

Rd+1:Rd ← Rd+1:Rd - K

Multiplication: MUL, MULS, MULSU

MUL: Multiply Unsigned

This instruction performs 8-bit × 8-bit → 16-bit unsigned multiplication.

R1:R0 ← Rd × Rr (unsigned ← unsigned × unsigned)

MULS: Multiply Signed

This instruction performs 8-bit × 8-bit → 16-bit signed multiplication

R1:R0 ← Rd × Rr (signed ← signed × signed)

MULSU: Multiply Signed with Unsigned

This instruction performs 8-bit × 8-bit → 16-bit multiplication of a signed and an unsigned number.

R1:R0 ← Rd × Rr (signed ← signed × unsigned)

Fractional Multiplication: FMUL, FMULS, FMULSU

FMUL: Fractional Multiply Unsigned

This instruction performs 8-bit × 8-bit → 16-bit unsigned multiplication and shifts the result one bit left

R1:R0 ← Rd × Rr (unsigned (1.15) ← unsigned (1.7) × unsigned (1.7))

FMULS: Fractional Multiply Signed

This instruction performs 8-bit × 8-bit → 16-bit signed multiplication and shifts the result one bit left.

R1:R0 ← Rd × Rr (signed (1.15) ← signed (1.7) × signed (1.7))

FMULSU: Fractional Multiply Signed with Unsigned

This instruction performs 8-bit × 8-bit → 16-bit signed multiplication and shifts the result one bit left.

R1:R0 ← Rd × Rr (signed (1.15) ← signed (1.7) × unsigned (1.7))

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 the low byte of the number stored in memory address 0x0110 into register r0

    load the high byte of the number stored in memory address 0x0111 into register r1

    load the low byte of the number stored in memory address 0x0120 into register r2

    load the high byte of the number stored in memory address 0x0121 into register r3

    add r0 and r2 together and store value in r0

    add with carry r1 and r3 and store value in r1

    store r1 into memory address 0x0101

    store r0 into memory address 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 the high byte of the number stored in memory address 0x0111 into register r0

    load the low byte of the number stored in memory address 0x0110 into register r1

    load the high byte of the number stored in memory address 0x0121 into register r2

    load the low byte of the number stored in memory address 0x0120 into register r3

    subtract r2 from r0 and store value into r0

    subtract with carry r3 from r1 and store value into r1

    store r0 into memory address 0x0101

    store r1 into memory address 0x0100

REFERENCE

http://ww1.microchip.com/downloads/en/devicedoc/atmel-0856-avr-instruction-set-manual.pdf