Zachary DeVita

ECE 375

Lab#5 PreLab

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.*

ADD – This adds two registers without the C flag and places the result in the destination register.

ADC – This adds two registers and the contents of the C flag and places the result in the destination register.

ADIW – This adds an immediate value to a register pair and places the result in the register pair.

SUB – This subtracts two registers and places the result in the destination register.

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

SBC – Subtracts two registers and subtracts with the C flag and places the result in the destination register.

SBCI – Subtracts a constant from a register and subtracts with the C flag and places the result in the destination register.

SBIW – Subtracts an immediate value from a register pair and places the result in the register pair.

AND – Performs the logical AND between the contents of register Rd and register Rr and places the result in the destination register.

ANDI – Performs the logical AND between the contents of register Rd and a constant and places the result in the destination register.

OR – Performs the logical OR between the contents of register Rd and register Rr and places the result in the destination register.

ORI – Performs the logical OR between the contents of register Rd and a constant and places the result in the destination register.

EOR – Performs the logical EOR between the contents of register Rd and register Rr and places the result in the destination register.

COM – This instruction performs a one's complement of register.

NEG – Replaces the contents of register Rd with its two's complement.

SBR – Sets specified bits in register Rd. Performs the logical ORI between the contents of register Rd and constant mask K and places the result in the destination register Rd.

CBR – Clears the specified bits in register Rd. Performs the logical AND between the contents of register Rd and the complement of the constant mask K. The result will be placed in register Rd.

INC – Adds one -1- to the contents of register Rd and places the result in the destination register Rd.

DEC – Subtracts one -1- from the contents of register Rd and places the result in the destination register Rd.

TST – Tests if a register is zero or negative. Performs a logical AND between a register and itself. The register will remain unchanged.

CLR – Clears a register. This instruction performs an Exclusive OR between a register and itself.

SER – Loads $FF directly to register Rd.

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

MULS – This instruction performs 8-bit × 8-bit → 16-bit signed multiplication.

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

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

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

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

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 the highest byte of a multibyte value is stored in the highest address of its range of addresses.)*

Load register R0 with $0110

Load register R1 with $0111

Load register R2 with $0120

Load register R3 with $0121

Add register R2 to register R0 (ADD R0, R2)

Add with carry register R3 to register R1 (ADC R1, R3)

Store the contents of register R0 to $0100

Store the contents of register R1 to $0101

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 register R0 with $0110

Load register R1 with $0111

Load register R2 with $0120

Load register R3 with $0121

Subtract register R0 from register R2 (SUB R0, R2)

Subtract with carry register R3 to register R1 (SBC R1, R3)

Store the contents of register R0 to $0100

Store the contents of register R1 to $0101