CEG 3136 – Computer Architecture II
Tutorial 1 – Basic Programming - Solution
Fall 2019

Questions:

1. What is the meaning of the sign bit = 1 when unsigned binary coded numbers are added?

   It indicates that the most significant bit is set to 1, which does not have any real meaning for unsigned numbers.

2. What is the meaning of the carry bit = 1 when two unsigned binary coded numbers are subtracted? When two's complement binary coded numbers are subtracted?

   When two unsigned numbers used in a subtraction, C=1 indicates that an overflow (or underflow) has occurred, i.e. the result of the subtraction is too large to be stored in the register containing the data operand (i.e. 8 or 16-bits –discuss when 8-bit subtraction occurs or 16-bit addition occurs, adding to the D register). What does it mean to large – the result is negative, i.e. a borrow was required to complete the subtraction.

   When two signed numbers are subtracted, C=1 has no significance.

3. What addressing mode is best to use when you want to access several sequential elements in a data array – immediate, direct, indexed?

   Indexed. Ask about function of the base address (used to provide the start address of the sequence of elements) and the offset (used to index into the sequence. For MC68HC12, where is the base address (in the index register) and where are the offsets found (an instruction operand). Offset can be what size – 5-bit, 9-bit or 16-bit offsets. Provide the general format of the indexed machine code instruction:

   Opcode Postbyte Offset Offset
   Where is the 5-bit offset found? In the postbyte.
   How many bytes in a machine code instruction with 9-bit offset? 3 bytes, The third byte + 1 bit in the postbyte provides the 9-bit offset.

4. Pointer register addressing with auto-increment and auto-decrement is referred to as what type of addressing for the MC68HC12?

    Indexed addressing with auto-increment and auto-decrement.
    What is meant by pointer addressing? The index register contains the address to the memory location used in the operation.
    What registers can be used as pointer registers in the CPU12? X, Y, SP, but not the PC?

Exercise 1

Give the values of the starting address, offset (indicate the size of the offset) and calculate the effective address of each of the following examples. Illustrate on the programmer's model the effect of the instructions to the CPU registers and memory.

a. X = $3000 A=$E5
    LDY A,X   EA = $30E5
b. Y = $6F40
    STD 4,-Y   EA=$6F3C
c. SP = $0BF3
    ADDA 3,SP   EA=$0BF6

Exercise 2:

An array of bytes contains a set of non-zero unsigned values. The last value in the array contains the value 0 to indicate the end of the array. Write a piece of code that contains a loop to read the contents of each byte into accumulator A. The array starts at address ARRAY. Write a first version that uses indexed addressing with accumulator B as the offset. Write a second version that auto-increments the index register.

```
; Accumulator B as offset
; This may be a bit challenging for students as B must be incremented before loading
; accumulator A for the branch to work since incb affects the Z bit.

        ARRAY EQU $4000

        ldx #ARRAY          ; Get start address of the array
        ldab #$FF           ; index <- -1
Loop    incb                ; increment index
        ldaa B,X            ; Get value
        bne Loop
```

```
; Indexed addressing with auto-incrementing

        ARRAY EQU $4000

        ldx #ARRAY
Loop    ldaa 1,X+
        bne Loop
```

Exercise 3

Consider two three byte numbers stored at addresses NUM1 and NUM2 as defined by the following assembler directives:

```
        NUM1        EQU         $0850
        NUM2        EQU         $0853
        DIFF        EQU         $0856
```

It is possible to load into the accumulators the bytes (or store into memory) using the labels as follows:

```
        LDAA        NUM1        ; Loads most significant byte into accumulator A
        STAB        DIFF+2      ; Store the contents of accumulator B into the least
                                ; significant byte of the DIFF number
```

The assembler will translate the labels to the appropriate addresses as shown below:

```
        LDAA        $0850       ; NUM1
        STAB        $0858       ; DIFF+2
```

The objective of this exercise is to write assembler source code to compute the difference between the two three-byte numbers (NUM1-NUM2) and store the result at address DIFF.

You will need the following 68HC12 instructions:

LDAA: Load accumulator A
SUBA: Subtract memory from A: A <- A- (M)
SBCA: Subtract memory and carry from A: A <- A – (M) - C
STAA: Store accumulator A

Use the programming model to follow the steps used in running the program.

```
; Subtract least significant byte
    LDAA NUM1+2
    SBCA NUM2+2
    STAA  DIFF+2
; Second byte
   LDAA NUM1+1
   SUBCA NUM2+1
   STAA DIFF+1
; Most significant byte
   LDAA NUM1
   SUBCA NUM2
   STAA DIFF
```