Problem Set #3 - Due 02/26/02

Total 100

Problem 1 - 10 points

To compute $x + y - z$,

1. Load $x$ in register R1
2. Load $y$ in register R2
3. Add $x$, $y$ and store the result in register R3
4. Load $z$ in register R4
5. Compute 2’s complement of $z$ and store it in register R5
6. Add 2’s complement of $z$ (in R5) to the result (in R3), store it in R3

To compute $2x + y$

1. Load $x$ into register R1
2. Load $y$ into register R2
3. There are two ways of computing $2x$
   a. Add $x$ to itself and store the result in R3
   b. Or Shift R1 left once and store the result in R3
4. Add $2x$ (in R3) to $y$ (in R2) and store it in R4

Problem 2 - 10 points

a. 1855
b. 2855
c. A403
d. 80F2
e. BB31

Problem 3 - 10 points

a. AND the byte with 1100 0011
b. XOR the byte with 1111 1111
c. XOR the byte with 1000 0000
d. OR the byte with 1000 0000
e. OR the byte with 0111 1111

Problem 4 - 10 points

1. 11010
2. 0000 1111
3. 001010
4. 9F
5. FF
Problem 5  - 30 points

1. To perform the subtraction operation of $A - B$, we need to compute the two’s complement of B and add it to A. i.e. $A - B = A + 2$’s complement of B
   - Load R1 with the first number
   - Load R2 with the second number
   - Obtain the 1’s complement of the second number by $R2 \ XOR \ 0xff$, and store the result in R3.
   - Add 1 to the 1’s complement to obtain the 2’s complement, store the result in R4.
   - Add R4, R1 and store the result in R5 to complete the subtraction algorithm.

2. [Diagram of the subtraction algorithm]
Program name: Subtraction using add only
Programmer: Jayakanth Srinivasan
Last Modified: Feb 18 2003

; code segment
load R1,1 ;1 added for computing 2's complement
load R2,0xff ;mask for flipping the bits
load R3, [first_number]; 'A' in A-B
load R4, [second_number]; 'B' in A-B
xor R5, R4, R2 ;flip the 0's and 1's in the second number
addi R6, R5, R1 ;add 1 to the flipped bits to get the 2's complement
addi R7, R6, R3 ;add the numbers to obtain A - B
load R8, 0xf2 ;address to store the result
store R7, [R8] ;store the results in memory location 0xf2
halt

; Data segment
definition first_number:
definition second_number:

Problem 6 - 30 points

1. The Shift and Add algorithm is used to implement the multiplication algorithm using only the add function and other logical operations. The algorithm is detailed below.

   a. Let the numbers be A, B
   b. Initialize the Product to 0
   c. Create bit_mask = 00000001b (used to check if the bit position is 1)
   d. Create shift_mask = 11111110b (used to set the last bit to zero for implementing the shift operation using rotate right)
   e. Repeat the following steps 8 times
      i. If the result of (B AND bit_mask) is non-zero
         - Product := Product + A;
      ii. Rotate A right seven times
      iii. A := A AND shift_mask (to set the last bit to zero)
      iv. Rotate bit_mask right seven times
   f. Store the Product in memory location 0xf2
Program to multiply two unsigned 8 bit integers
Programmer: Jayakanth Srinivasan
Date Last Modified: Feb 20th 2003
Constraints: The program will not work for numbers
whose product is greater than 255

code segment

load R0, 0;
load R1, [bit_mask] ; loads the bit-mask used to determine
                    ; if addition is necessary
load R2, [shift_mask] ; mask to set the last bit to zero
load R3, [first_number]; A in A*B
load R4, [second_number]; B in A*B
load R5, 0 ; register used to store the product

multiply_loop:
    and R6, R4, R1;
    jmpEQ R6=R0, dont_add;
    addi R5, R5, R3 ; add the first number to the product
dont_add:
    ror R3, 7 ; get a 1 bit position left shift
    and R3, R3, R2 ; complete the shift by inserting a 0
    ror R1, 7 ; shift bit_mask left 1 position
    and R6, R1, R2 ; and to see if the bit_mask has
                    ; cycled back to the lsb
    jmpEQ R6=R0, done ; jmp if you have looped 8 times
    jmp multiply_loop ; repeat the loop otherwise
done:
    store R5, [0xf2] ; store the result in location 0xf2

data segment

bit_mask:     db 00000001b;
shift_mask:   db 11111110b;
first_number: db 8d;
second_number: db 8d.