ΗΡΥ 201– Ψηφιακοί Υπολογιστές

Γ. Παπαευσταθίου

Ασκήσεις

Exercise 12

Write a MIPS assembly language function that accepts a binary number in register $a0 and returns a value corresponding to the number of one’s in the binary number.

Exercise 12(Pseudocode)

```
$v0 = 0;
while ($a0 = !0)
    {
    $t0 = $a0 & 1;
    $a0 = $a0 >> 1;
    $v0 = $v0 + $t0;
    }
Return
```

Exercise 12 (MIPS Assembly Language)

```
Label   Op-Code Dest. S1, S2   Comments
        .globl  count
        .text
count:
        li      $v0, 0
while:
        andi    $t0, $a0, 1
        srl     $a0, $a0, 1
        add     $v0, $v0, $t0
        bnez    $a0, while
        jr      $ra
```

Exercise 13

Translate the following pseudocode expression to MIPS assembly language code. Include code to insure that there is no array bounds violation when the store word (sw) instruction is executed. Note that the array “zap” is an array containing 50 words, thus the value in register $a0 must be in the range from 0 to 196. Include code to insure that the value in register $a0 is a word address offset into the array “zap.” If an array bounds violation is detected or the value in register $a0 is not a word address offset then branch to the label “Error.”

```
        .data
zap:    .space      200
        .text
        . . .
        zap[$a0] = $s0
```

Exercise 13 (Pseudocode)

```
$t0 = $a0 & 3;
If ($t0 != 0 )   go to Error;
if ($a0 < 0)     go to Error
if ($a0 > 196)   go to Error
$t0 = &zap
$a0 = $a0 + $t0
Mem($a0) = $s0;
```

Exercise 13 (MIPS Assembly Language)

```
Label  Op-Code  Dest. S1, S2     Comments
       .data
zap:   .space        200
       .text
       andi     $t0, $a0, 3
       bnez     $t0, Error
       bltz     $a0, Error
       li       $t0, 196
       bgt      $a0, $t0, Error
       la       $t0, zap
       add      $a0, $a0, $t0
       sw       $s0, 0($a0)
```

Exercise 14

Write a function to search through an array “X” of “N” words to find how many of the values are evenly divisible by four. The address of the array will be passed to the function using register $a0, and the number of words in the array will be passed in register $a1. Return the results in register $v0.

Exercise 14 (Pseudocode)

```
$v0 = 0;
$t3 = 3;
For (  ;   $a1 > 0;   $a1= $a1- 1)
    { $t2 = Mem ($a0);
    $a0 = $a0 + 4;
    $t0 = $t2 &  t3;
    If ($t0 == 0 )   $v0 = $v0 + 1;
    }
    return
```

Exercise 14 (MIPS Assembly Language)

```
Label      Op-Code Dest. S1, S2
           Comments
Div4:
           li         $v0, 0
           li         $t3, 3
           b          skip
loop:
           lw         $t2, 0($a0)
           addi       $a0, $a0, 4
           and        $t0, $t2, $t3
           bnez       $t0, skip
           addi       $v0, $v0, 1
skip:
           addi       $a1, $a1, -1
           bgez       $a1, loop
           jr         $ra
```

Exercise 15

MinMax (&X, N, Min, Max)

Write a function to search through an array 'X' of 'N' words to find the minimum and maximum values.

The parameters &X and N are passed to the function on the stack, and the minimum and maximum values are returned on the stack. (Show how MinMax is called)

Exercise 15 (Pseudocode)

```
                MinMax(&X:$t1,  N:$t2, min:$t8,  max:$t9)
$t1 = Mem($sp);
$t2 = Mem($sp+4);
$t8 = Mem($t1);
$t9 = $t8;
$t2 = $t2 - 1;
While ($t2 > 0)
    {$t1 = $t1 + 4;
    $t0 = Mem($t1);
    if ($t0 < $t8)  $t8 = $t0;
    else  if ($t0 > $t9)  $t9 = $t0;
    $t2= $t2 - 1;
    }
Mem($sp+8) = $t8;
Mem($sp+12) = $t9;
```

Exercise 15 (MIPS Assembly Language)

```
Label     Op-Code Dest. S1, S2     Comments
##### An Example of calling the function  #####
          .data
array     .space  400
          .text
          addiu   $sp,      $sp,  -16
          la      $t0,      array
          sw      $t0,      0($sp)
          li      $t0,      100
          sw      $t0,      4($sp)
          jal     MinMax
          lw      $t0,      8($sp)
          lw      $t1,      12($sp)
          addiu   $sp,      $sp,  16
```

Exercise 15 **MinMax(&X:$t1, N:$t2, min:$t8, max:$t9)**

Label *Op-Code* *Dest. S1, S2* *Comments*

```
        .text
MinMax:
        lw      $t1, 0($sp)             # get &X
        lw      $t2, 4($sp)             # get N
        lw      $t8, 0($t1)             # Init. min
        move    $t9, $t8                # Init. max
        addi    $t2, $t2, -1
        blez    $t2, ret
loop:
        addiu   $t1, $t1, 4
        lw      $t0, 0($t1)
        bge     $t0, $t8, next
        move    $t8, $t0
        b       chk
next:
        ble     $t0, $t9, chk
```

Exercise 16

Search(&X, N, V, L)

Write a function to sequentially search an array X of N **bytes** for the relative location L of a value V.

The parameters &X, N, and V are passed to the procedure on the stack, and the relative location L

(a number ranging from 1 to N) is returned on the stack.

If the value V is not found the value -1 is returned for L.

Exercise 16 (Pseudocode)

```
    $t3= Mem(sp);                          # get &X
    $t1= Mem($sp + 4);                     # get N
    $t0= Mem($sp + 8);                     # get V
    $t2=$t1;
    for ($t2 = $t2 - 1; $t2 >= 0;  $t2= $t2 - 1)
    { $t4 = mem($t3);
     $t3=$t3 + 1;
    if ( $t4 ==  $t0) go to found;
    }
    Mem(sp + 12) = -1;
    go to exit;
found:
    Mem(sp + 12) = $t1- $t2;
exit:           return;
```

Exercise 16 (MIPS Assembly Language)

```
Label      Op-Code Dest.      S1, S2    Comments
           .text
search:
           lw      $t3,       0($sp)    # get &X
           lw      $t1,       4($sp)    # get N
           lw      $t0,       8($sp)    # get V
           move    $t2,       $t1
           addi    $t2,       $t2, -1   # t2 = N - 1
loop:
           lbu     $t4,       0($t3)    # get a character
           addiu   $t3,       $t3, 1    # increment pointer
           beq     $t4,       $t0, found
           addi    $t2,       $t2, -1   # decrement loop counter
           bgez    $t2,       loop
           li      $t4,       -1
           sw      $t4,       12($sp)
           b       exit
found:
           sub     $t1,       $t1, $t2
           sw      $t1,       12($sp)
exit:      jr      $ra
```

Exercise 17

Scan(&X, N, U, L, D)
Write a function to scan an array 'X' of 'N' bytes counting how many bytes are ASCII codes for:

a. upper case letters - U
b. lower case letters - L
c. decimal digits - D

Return the counts on the stack. The address of the array and the number of bytes N will be passed to the function on the stack.

Write a short main program to test this function.

A Main Program to Test the Scan Function

```
        .data
string: .asciiz “The Quick Fox  0123456789”
        .text
main:   -----
        -----
        addiu   $sp,    $sp, -20                # Allocate
        la      $t0,    string
        sw      $t0,    0($sp)
        li      $t0,    24
        sw      $t0,    4($sp)

        jal     Scan

        lw      $t0,    8($sp)
        lw      $t1,    12($sp)
        lw      $t2,    16($sp)
        addi    $sp,    $sp, 20         # Deallocate
                ----
```

Exercise 17 (Pseudocode)

```
Scan(&X:$t6, N:$t2, U:$t3, L:$t4, D:$t5)
$t6 = Mem(sp)                    #  &X
$t2 = Mem(sp+4)                  #   N
$t3=$t4=$t5=0;
For ( ; $t2> 0;  $t2=$t2-1)
{
$t1 = mem($t6)                   # get a byte
$t6 = $t6 + 1
if ( $t1 >= 65 && $t1 <= 90  )     $t3 = $t3+1;
else if ( $t1 >= 97 && $t1 <= 122) $t4=$t4+1;
else if ( $t1 >= 48 && $t1 <= 57  ) $t5=$t5+1;
}
Mem(sp + 8 ) = $t3;
Mem(sp + 12 ) = $t4;
Mem(sp + 16 ) = $t5;
return;
```

Exercise 17 (Assembly Language Initialize)

Scan(&X:$t6, N:$t2, U:$t3, L:$t4, D:$t5)

```
Label   Op-Code Dest. S1, S2     Comments
scan:
        lw      $t6,  0($sp)     # Get &X
        lw      $t2,  4($sp)     # Get Value N
        li      $t3,  0          # Count of Upper Case
        li      $t4,  0          # Count of Lower Case
        li      $t5,  0          # Count of Decimal Digits
        blez    $t2,  done
        li      $t0,  48         # ASCII “0”
        li      $t9, 57          # ASCII  “9”
        li      $t7, 97          # ASCII  “a”
        li      $t8, 122         # ASCII  “z”
        addiu   $sp, $sp, -8     # Allocate Temp Space
        sw      $s6, 0($sp)      # Save s6
        sw      $s7, 4($sp)      # Save s7
        li      $s6, 65          # ASCII “A”
        li      $s7, 90          # ASCII “Z”
```

Exercise 17 (Assembly Language Body)

```
Label     Op-Code Dest. S1, S2          Comments
loop:     lbu     $t1,  0($t6)
          addi    $t6,  $t6, 1
          blt     $t1,  $s6, num        # “A”
          bgt     $t1,  $s7, lowc       # “Z”
          addi    $t3,  $t3, 1
          b       check
lowc:
          blt     $t1,  $t7,   check    # “a”
          bgt     $t1,  $t8, check      # “z”
          addi    $t4,  $t4, 1
          b       check
num:
          blt     $t1,  $t0, check      # “0”
          bgt     $t1,  $t9, check      # “9”
          addi    $t5,  $t5, 1
check:
          addi    $t2,  $t2, -1
          bgtz    $t2,  loop
```

Exercise 17 (Assembly Language Continued)

```
Label     Op-Code  Dest. S1, S2        Comments


          lw       $s6, 0($sp)         # Restore s6
          lw       $s7, 4($sp)         # Restore s7
          addiu    $sp, $sp, -8        # Deallocate Temp Space
          sw       $t3,  8($sp)
          sw       $t4,  12($sp)
          sw       $t5,  16($sp)
          jr       $ra
```

Exercise 18

Hypotenuse(A, B, H)

This is an exercise in calling nested functions and passing parameters on the stack.

Write a function to find the length of the hypotenuse of a right triangle whose sides are of length A and B. Assume that a math library function “sqr (V, R)” is available, which will compute the square root of any positive value V, and return the square root result R.

Write a main program to test this function.

A Main Program to test hypotenuse

```
main:
            addi        $sp,   $sp, -12        # allocate
            li          $t0,   3
            sw          $t0,   0($sp)
            li          $t0,   4
            sw          $t0,   4($sp)

            jal         hypotenuse

            lw          $a0,   8($sp)          # get result
            addi        $sp,   $sp, 12              # deallocate
            li          $v0,   1               # print result
            syscall
            li          $v0,   10
            syscall
```

Exercise 18 (Pseudocode)

```
t0 = Mem(sp);
t1 = Mem(sp+4);
Mem(sp+8) = sqr (  t0*t0  +  t1*t1  ) ;
return
```

Exercise 18 (Assembly Language)

```
Label       Op-Code  Dest. S1, S2       Comments
hypotenuse:
            lw       $t0, 0($sp)        # Get A
            lw       $t1, 4($sp)        # Get B
            mult     $t0, $t0
            mflo     $t0
            mult     $t1, $t1
            mflo     $t1
            add      $t0, $t0, $t1
            addi     $sp,$sp, -12       # Allocate
            sw       $t0, 0($sp)        # Pass Value to sqr
            sw       $ra, 8($sp)        # Save ra
            jal      sqr                # Call sqr
            lw       $t0, 4($sp)        # Get sqr Result
            lw       $ra, 8($sp)        # Restore ra
            addi     $sp, $sp, 12       # Deallocate
            sw       $t0, 8($sp)        # Return Hypotenuse
            jr       $ra
```

Exercise 19

AVA (&X, &Y, &Z, N, status)

Write a function to perform an absolute value vector addition.
Use the stack to pass parameters. The parameters are:
the starting address of three different word arrays (vectors) : X, Y, Z,
and an integer value N specifying the size of the vectors.

If overflow ever occurs when executing this function,
an error status of “1” should be returned and the function aborts
any further processing. Otherwise, return the value “0” for status.
The function will perform the vector addition:

$Xi = | Yi | + | Zi |$; with i going from 0 to $N - 1$.

Also write a main program to test this function.

Example code for testing the AVA function

```
Label   Op-Code Dest. S1, S2        Comments
        `.data
zig:    .space        20
zag:    .word 345, 765, -234567, 2345, 999
zonk:   .word -38645, 765987, 67, 3215, 444
msg:    .asciiz       "Overflow Occurred"
        .text
main:   addi   $sp,  $sp, -20          # Allocate
        la     $s0,  zig
        sw     $s0,  0($sp)
        la     $s0,  zag
        sw     $s0,  4($sp)
        la     $s0,  zonk
        sw     $s0,  8($sp)
        li     $s0,  5
        sw     $s0,  12($sp)
        jal    AVA
        lw     $s0,  16($sp)
```

Exercise 19 (Pseudocode)

```
AVA(&X:$t6,  &Y:$t7,  &Z:$t8, N:$t0, status)
$t6 = Mem($sp);
$t7 = Mem($sp+4);
$t8 = Mem($sp+8);
$t0= Mem ($sp+12);
for (   ; $t0  > 0 ; $t0 = $t0 - 1)
              { $t1= Mem($t7);
                $t7 = $t7 + 4;
                if ($t1 < 0) $t1 =  0 - $t1;
                $t2= Mem($t8);
                $t8 = $t8 + 4;
                if ($t2 < 0) $t2 = 0 - $t2;
                $t1 = $t1 + $t2;
                if ($t1< 0) go to ovf;
                Mem($t6) = $t1;
               $t6 = $t6 + 4;
              }
Mem($sp+16) = 0;
return
ovf:  Mem($sp+16) = 1;  return
```

Exercise 19 (Assembly Language)

```
Label   Op-Code Dest. S1, S2    Comments
                .text
AVA:    lw      $t6,    0($sp)  # Get &X
        lw      $t7,    4($sp)  # Get &Y
        lw      $t8,    8($sp)  # Get &Z
        lw      $t0,    12($sp)         # Get N
loop:   lw      $t1,    0($t7)
        addiu   $t7,    $t7, 4
        bgez    $t1,    next
        sub     $t1,    $0, $t1
next:   lw      $t2,    0($t8)
        addiu   $t8,    $t8, 4
        bgez    $t2,    sum
        sub     $t2,    $0, $t2
sum:    add     $t1,   $t1,     $t2
        bltz    $t1,    ovf
        sw      $t1,    0($t6)
        addiu   $t6,    $t6, 4
        addi    $t0,    $t0, -1
```

Exercise 20

Fibonacci (N, E)
Write an function to return the Nth element in the Fibonacci sequence.
A value N is passed to the function on the stack, and the Nth Fibonacci number E is returned on the stack.

If N is greater than 46 overflow will occur, so return a value of 0 if N is greater than 46. Also show an example of calling this function to return the 10th element in the sequence.

The first few numbers in the Fibonacci sequence are: 0, 1, 1, 2, 3, 5 . . . .

Example Code to Test Fibonacci

```
Label    Op-Code  Dest. S1, S2      Comments
         .data
msg:     .asciiz  “Can not Compute Correct Result”
         .text
main:    addi     $sp,  $sp, -8     # Allocate
         li       $t0,  10          # Pass argument to Fib
         sw       $t0,  0($sp)
         jal      Fib               # Call Fibonacci
         lw       $a0,  4($sp)      # Get result back
         addi     $sp,  $sp, 8      # Deallocate
         bgtz     $a0,  done
         li       $v0,  4
         la       $a0,  msg
         syscall
done:    li       $v0,  1           # Print result
         syscall
         li       $v0,  10
         syscall
```

Exercise 20 (Pseudocode)

```
$t0 = Mem($sp);
if ($t0 > 46) {Mem($sp+4) = 0; Return;}
  If ($t0 > 1)
          {$t1 = 0;    $t2 = 1;
                     For ($t0 = $t0 - 1; $t0 > 0;  $t0 = $t0 - 1)
                     {
                     $t3 = $t2 + $t1;
                     $t1= $t2;
                     $t2 = $t3
                     }
          }
  else  $t3= $t0;
Mem($sp+4) = $t3;
Return
```

Exercise 20 (Fibonacci Assembly Language)

```
Label       Op-Code Dest. S1, S2      Comments
Fib:
            lw        $t0,  0($sp)        # Get N
            bltz      $t0,  error
            addi      $t1, $t0, -46
            bgtz      $t1, error
            li        $t1,  0
            li        $t2,  1
            move      $t3, $t0
            addi      $t0, $t0,  -1
            blez      $t0, done
loop:
            add       $t3,  $t2, $t1
            move      $t1,  $t2,
            move      $t2,  $t3
            addi      $t0,  $t0, -1
            bgtz      $t0,  loop
done:
            sw        $t3,  4($sp)        # Return Nth Fibonacci number
            jr        $ra
error:
            sw        $0, 4($sp)
            jr        $ra
```

Fibonacci (A Very Efficient Method)

```
        .data
fibnum: .word   0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,
        .word   6765,10946,17711,28657,46368,75025,121393,196418,317811,514229,
        .word   832040,1346269,2178309,3524578,5702887,9227465,14930352,24157817,
        .word   39088169,63245986,102334155,165580141,267914296,
        .word   433494437,701408733,1134903170,1836311903
        .text
fib:
        lw      $t0, 0($sp)
        bltz    $t0, error
        addi    $t1, $t0, -46
        bgtz    $t1, error
        sll     $t0, $t0, 2
        la      $t1, fibnum
        addu    $t0, $t1, $t0
        lw      $t0, 0($t0)
        sw      $t0, 4($sp)
        jr      $ra
error:
        sw      $0, 4($sp)
        jr      $ra
```

Exercise 21

<u>BubSort (&X, N)</u>
Write a function to sort an array ‘“ X ” of ‘“N” words into ascending order using the bubble sort algorithm.

The address of the array and the value N will be passed to the function on the stack.

Show how the sort function is called.

Example Assembly Language Code to Call Sort(&Z, 1000)

```
addi    $sp,    $sp, -8
la      $t0,    z
sw      $t0,    0($sp)
li      $t0,    1000
sw      $t0,    4($sp)
jal     sort
addi    $sp,    $sp, 8
```

Exercise 21 (Pseudocode)

```
BubSort (&X:$t3, N:$t0)
        $t0 = Mem($sp+4);
Again:
        $t0 = $t0 - 1;
        $t2=0;
        $t3= Mem($sp);
        For ($t1= $t0;  $t1 > 0;  $t1 = $t1-1)
                {If ( Mem($t3)  >  Mem($t3+4)  ) then
                        {exchange Mem($t3) & Mem($t3+4)
                        $t2=1}
                $t3= $t3 +4;
                }
        If ($t2 == 1) go to Again
        else
        return
```

Exercise 21 (Assembly Language)

```
Label     Op-Code Dest. S1, S2     Comments
BubSort:
          lw      $t0,  4($sp)     # Get N
again:
          addi    $t0,  $t0, -1
          li      $t2,  0          # Clear flag
          lw      $t3,  0($sp)     # Get pointer to array
          move    $t1,  $t0,                 # Init. loop count
loop:
          lw      $t8,  0($t3)
          lw      $t9,  4($t3)
          ble     $t8,  $t9, next
          sw      $t8,  4($t3)     # Swap values
          sw      $t9,  0($t3)
          li      $t2,  1          # Set Flag
next:
          addi    $t3,  $t3, 4               # Inc. pointer
```