ΗΡΥ 201– Ψηφιακοί Υπολογιστές

Γ. Παπαευσταθίου

Ασκήσεις

Exercise 1

Write a MIPS assembly language program to find the Sum of the first 100 words of data in the memory data segment with the label “chico”. Store the resulting sum in the next memory location beyond the end of the array chico.

Exercise 1 (Pseudo Code)

```
$a0 = &chico;    # “&” means “Address of”
$t0 = 0;
For ($t1= 100;   $t1 > 0;   $t1= $t1- 1)
   {
   $t0 = $t0 + Mem($a0);
   $a0 = $a0 + 4;
   }
Mem($a0) = $t0;
```

Exercise 1 (MIPS Assembly Language)

```
Label     Op-Code   Dest. S1, S2        Comments
          .data
chico:    .space    400
result:   .word
          .globl    main
          .text
main:
          la        $a0, chico          # Load address pointer
          li        $t0, 0              # Clear sum
          li        $t1, 100            # Initialize loop count
loop:
          lw        $t2, 0($a0)         # $t2 = Mem(a0)
          add       $t0, $t0, $t2       # $t0 = $t0 + $t2
          addi      $a0, $a0, 4         # Inc. address pointer
          addi      $t1, $t1, -1        # Dec. loop count
          bgtz      $t1, loop                   # if ($t1 > 0) branch
          sw        $t0, 0($a0)         # Store the result
          li        $v0, 10             # End of program
          syscall
```

Exercise 2

Write an efficient segment of MIPS assembly language code to transfer a block of 100 words starting at memory location “SRC” to another area of memory beginning at memory location “DEST”.

Exercise 2 (Pseudo Code)

```
$a1= &SRC;  # “&” means “Address of”
$a2= &DEST;
for ($t0 = 100; $t0 > 0; $t0 =$t0 -1)
    {$t1 = Mem($a1);
     Mem($a2) = $t1;
      $a1= $a1 + 4;
      $a2= $a2 + 4;
   }
```

Exercise 2 (MIPS Assembly Language)

```
Label   Op-Code  Dest. S1, S2        Comments
        .data
SRC:    .space   400
DEST:   .space   400
        .globl   main
        .text
main:
        la       $a1, SRC           # $a1 = &SRC
        la       $a2, DEST          #$a2 = &DEST
        li       $t0, 100           #$t0 = 100
loop:   lw       $t1, 0($a1)        #$t1= Mem($a1)
        sw       $t1, 0($a2)        #Mem($a2) = $t1
        addi     $a1, $a1,4         #$a1 = $a1+4
        addi     $a2, $a2,4         #$a2 = $a2+4
        addi     $t0, $t0, -1       # $t0 = $t0 - 1
        bgtz     $t0, loop                  #Branch if $t0 > 0
        li       $v0, 10
        syscall
```

Exercise 3

Write a MIPS function which accepts an integer word in register $a0 and returns its absolute value in $a0.

Also show an example code segment that calls the ABS function twice, to test the function.

Exercise 3 (Pseudo Code)

```
Function ABS($a0);
if ($a0 < 0) $a0 = $0 - $a0;
return;
```

Exercise 3 (MIPS Assembly Language)

```
Label     Op-Code  Dest. S1, S2        Comments
          .text
ABS:      bgez     $a0, return         # If ($a0 >= 0) done
          sub      $a0, $0, $a0        # $a0 = 0 - $a0
return:   jr       $ra                 #Return
##########################################
          .globl   main
          .text
main:     li       $a0, -9876
          jal      ABS
          li       $v0, 1              # Output result
          syscall
          li       $a0, 9876
          jal      ABS
          li       $v0, 1              # Output result
          syscall
          li       $v0,10              # End of program
          syscall
```

Exercise 4

Write a function PENO (&X, N, SP, SN) that will find the sum of the positive and negative values in an array X of length “N”.

"X" the address of an array, passed through $a0.

"N" is the length of the array, passed through $a1.

The procedure should return two values:

(1) The sum of all the positive elements in the array, passed back through $v0.

(2) The sum of all the negative elements in the array, passed back through $v1.

Exercise 4 (Pseudo Code)

```
$v0 = 0;
$v1 = 0;
for (  ;  $a1 > 0;  $a1 = $a1-1)
   {$t0 = Mem($a0);
   $t1 = $t0 & 1;
   $a0 = $a0 + 4;
   if ($t0  >  0 & $t1 = 0)  $v0 = $v0 +
   $t0;
   if ($t0  <  0 & $t1 != 0)   $v1= $v1+
   $t0;
   }
return;
```

Exercise 4 (MIPS Assembly Language)

Label	*Op-Code*	*Dest. S1, S2*	*Comments*
	.globl	**SUM**	
	.text		
PENO:			
	li	**$v0, 0**	
	li	**$v1, 0**	
LOOP:			
	lw	**$t0, 0($a0)**	
	andi	**$t2, $t0, 1**	
	addi	**$a0, $a0, 4**	
	bltz	**$t0, NEG**	
	bnez	**$t2, CHK**	
	add	**$v0, $v0, $t0**	
	b	**CHK**	
NEG:			
	beqz	**$t2, CHK**	
	add	**$v1, $v1, $t0**	
CHK:			
	addi	**$a1, $a1, -1**	
	bgtz	**$a1, LOOP**	
	jr	**$ra**	

Exercise 5

Write a function SUM(N) to find the sum of the integers from 1 to N, making use the multiplication and shifting operations. The value N will be passed to the procedure in $a0 and the result will be returned in the $v0 register.

Write a MIPS assembly language main program that will call the Sum function five times each time passing a different value to the function for N, and printing the results. The values for N are defined below:

```
                .data
N:      .word   9, 10, 32666, 32777, 654321
```

Exercise 5 (Pseudo Code)

```
Function SUM (a0: input value, $v0: output value)
$v0 = $a0 + 1;
$v0 = $v0 * $a0;
$v0 = $v0 >> 1                          #$v0 / 2;
return;
```

Exercise 5 (MIPS Assembly Language)

```
Label    Op-Code Dest. S1, S2        Comments
         .text
SUM:
         addi  $v0,  $a0, 1                 # $v0 = $a0 + 1
         mult  $v0,  $a0            # $v0 = $v0 * $a0
         mflo  $v0
         sra   $v0,  $v0, 1         # Shift right arithmetic
                                    # is the quick way to
                                    # divide by 2
         jr          $ra
```

The Main Program

```
                .data
N:              .word   9, 10, 32666, 32777, 654321
                .text
main:           li      $s0, 5
                la      $s1, N
loop:
                lw      $a0, 0($s1)
                addiu   $s1, $s1, 4
                jal     SUM
                move    $a0, $v0
                li      $v0, 1
                syscall
                addi    $s0, $s0, -1
                bnez    $s0, loop
                li      $v0, 10
                syscall
```

Exercise 6

Write a function FIB(N, &array) to store the First N elements of the Fibonacci sequence into an array in memory. The value N is passed in $a0, and the address of the array is passed in register $a1.

The first few numbers of the Fibonacci sequence are: 1, 1, 2, 3, 5, 8, 13, ...

Exercise 6 (Pseudo Code)

```
Mem($a1) = 1;
Mem($a1 + 4) = 1;
for ($a0 = $a0 - 2; $a0 > 0; $a0 = $a0-1)
    {
    Mem($a1+8) = Mem($a1) + Mem($a1+4);
    $a1 = $a1 + 4;}
return;
```

Exercise 6 (MIPS Assembly Language)

Label	*Op-Code*	*Dest. S1, S2*	*Comments*
fib:	**li**	**$t0, 1**	
	sw	**$t0, 0($a1)**	
	sw	**$t0, 4($a1)**	
	addi	**$a0, $a0, -2**	
loop:			
	lw	**$t0, 0($a1)**	
	lw	**$t1, 4($a1)**	
	add	**$t0, $t0, $t1**	
	sw	**$t0, 8($a1)**	
	addi	**$a1, $a1, 4**	
	addi	**$a0, $a0, -1**	
	bgtz	**$a0, loop**	
	jr	**$ra**	

Exercise 7

Write a function that receives 3 integer words in registers $a0, $a1, & $a2, and returns them in ordered form with the minimum value in $a0 and the maximum value in $a2.

Exercise 7 (Pseudo Code)

Function Order($a0,$a1,$a2);

If ($a0 > $a1) exchange $a0 and $a1;

if ($a1 > $a2) exchange $a1 and $a2 else return;

If ($a0 > $a1) exchange $a0 and $a1;

return;.

Exercise 7 (MIPS Assembly Language)

```
Label     Op-Code  Dest. S1, S2        Comments
          .text
order:
          ble      $a0, $a1, next
          move     $t0, $a1
          move     $a1, $a0
          move     $a0, $t0
next:
          ble      $a1, $a2, done
          move     $t0, $a2
          move     $a2, $a1
          move     $a1, $t0

          ble      $a0, $a1, done
          move     $t0, $a1
          move     $a1, $a0
          move     $a0, $t0
done:     jr       $ra
```

Exercise 8

Write the complete assembly language program,including data declarations, that corresponds to the following C code fragment.

Make use of the fact that multiplication and division by powers of 2 can be performed most efficiently by shifting.

```
int main()
{  int  K, Y ;
   int  Z[50] ;
   Y = 56 ;
   K = 20 ;
   Z[K] = Y  -  16  *  ( K / 4  + 210) ;
}
```

Exercise 8 (MIPS Assembly Language)

```
Label    Op-Code   Dest. S1, S2          Comments
         .globl    main
         .data
K:       .space    4
Y:       .space    1
Z:       .space    50
         .text
main:    la        t3, K
         li        $t0,  56
         sw        $t0,  4($t3)                  # Y= 56
         li        $t1,  20
         sw        $t1,  0($t3)                  # K= 20
         sra       $t1, $t1, 2           #  K/4
         addi      $t1, $t1, 210          #  K/4 + 210
         sll       $t1, $t1, 4           #   ---  x 16
         sub       $t2, $t0, $t1         # t2= Y – 16 * (K / 4 + 210)
         lw        $t1, 0($t3)                   # t1=K
         sll       $t1, $t1, 2           # scale K
         addu      $t1, $t1, $t3         # t1= & Z[k] - 8
         sw        $t2, 8($t1)                   # Z[K]= Y-16*(k/4+210)
```

Exercise 9

Write a function to search through an array “X” of “N” words to find the minimum and maximum values. 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. The minimum and maximum values are returned in registers $v0, & $v1.

Exercise 9 (Pseudo Code)

```
MaxMin ($a0: address, $a1: number of words)
$v0 = Mem($a0);
$v1 = $v0;
$a1 = $a1 - 1;
While ($a1 > 0)
   {$a0 = $a0 + 4;
   $t0 = Mem($a0);
   if ($t0 < $v0)  $v0 = $t0;
   else  if ($t0 > $v1)  $v1 = $t0;
   $a1= $a1 - 1;
   }
return;
```

Exercise 9 (MIPS Assembly Language)

```
Label     Op-Code  Dest. S1, S2          Comments
MaxMin:
          lw       $v0, 0($a0)
          move     $v1, $v0
          addi     $a1, $a1, -1
          blez     $a1, ret
loop:
          addi     $a0, $a0, 4
          lw       $t0, 0($a0)
          bge      $t0, $v0, next
          move     $v0, $t0
          b        chk
next:
          ble      $t0, $v1, chk
          move     $v1, $t0
chk:
          addi     $a1, $a1, -1
          bgtz     $a1, loop
ret:
          jr       $ra
```

Exercise 10

Write a function to find the sum of the main diagonal elements in a two dimensional N by N array of 32 bit words. The address of the array and the size N are passed to the procedure in registers $a0 and $a1 respectively.

The result is returned in $v0.

The values in registers $a0 and $a1 should not be modified by this procedure.

Calculate the number of clock cycles required to execute your algorithm, assuming N=4

Exercise 10 (Pseudocode)

```
$v0 = Mem($a0);
$t1 = $a0;
$t3=($a1+1) * 4;
for ($t0 = $a1-1; $t0 > 0, $t0= $t0-1)
{$t1= $t1+ $t3;
$v0= $v0 + Mem($t1) }
return
```

Exercise 10 (MIPS Assembly Language)

```
Label   Op-Code Dest. S1, S2    Comments
        .text
mdsum:
        lw      $v0, 0($a0)     # v0 = first element
        move    $t1, $a0
        addi    $t3, $a1, 1     # compute offset
        sll     $t3, $t3, 2     # multiply by 4
        addi    $t0, $a1, -1    # init. loop count
        blez    $t0, return
loop:   add     $t1, $t1, $t3   # calc. next address
        lw      $t2, 0($t1)     # t2=Mem(t1)
        add     $v0, $v0, $t2   # add to sum
        addi    $t0, $t0, -1    # decrement loop count
        bgtz    $t0, loop
return: jr      $ra
```

Exercise 11

Write a function to find the determinant of a two by two matrix (array). The address of the array is passed to the function in registers $a0 and the result is returned in $v0. The value in register $a0 should not be modified by this function.

Calculate the number of clock ticks required to execute your algorithm.

Exercise 11 (Pseudocode)

```
$v0 = Mem($a0) * Mem($a0+12) -
        Mem($a0+4) * Mem($a0+8);
Return
```

Exercise 11 (MIPS Assembly Language)

```
Label     Op-Code   Dest. S1, S2      Comments
          .globl    determ2
          .text
determ2:
          lw        $t0, 0($a0)
          lw        $t1, 12($a0)
          mult      $t1, $t0
          mflo      $v0
          lw        $t0, 4($a0)
          lw        $t1, 8($a0)
          mult      $t1, $t0
          mflo      $t0
          sub       $v0, $v0, $t0
return:
          jr        $ra
```