Arquitectura de Computadores I

Ano Lectivo 2013/14 - 1º Semestre

Mini-Teste Prático 2 – Terça-Feira, 10/12/2013, 9:00-11:00

Nº Mec.: _____ Nome: _____ PROPOSTA DE RESOLUÇÃO _____ Turma: _____

NOTE BEM: Leia atentamente todas as questões, comente o código usando a linguagem C e respeite a convenção de passagem de parâmetros e salvaguarda de registos que estudou. Na tradução para o *Assembly* do MIPS respeite rigorosamente os aspectos estruturais e a sequência de instruções indicadas no código original fornecido.
O código em C apresentado pode não estar funcionalmente correcto, pelo que **não deve ser interpretado**.

1) Codifique em *assembly* do MIPS a seguinte função `main`:

```
int atoi_t(char *, int);
float mean(float *, int);

int main(int argc, char *argv[])
{
    static float array[10];
    float *p = array;
    int i = 0;

    if (argc < 1) return -1;

    while ( (i < argc) && (i < 10) )
    {
       *p = (float)atoi_t(argv[i++], 10);
       p++;
    }

    print_float(mean(array, argc));//syscall
    return 0;
}
```

Label	Instrução em *assembly*	Comentário em C

```
      .data
array:.space 40 # static float array[10];

      .text
# argc  -> $a0,$s0
# argv  -> $a1,$s1
# p     -> $s2
# i     -> $s3

      .globl main
main: subu  $sp,$sp,20
      sw    $s0,0($sp)
      sw    $s1,4($sp)
      sw    $s2,8($sp)
      sw    $s3,12($sp)
      sw    $ra,16($sp)

      move  $s0,$a0     # s0 = argc;
      move  $s1,$a1     # s1 = argv;
      la    $s2, array  # p = array;
      li    $s3,0       # i=0;

if:   bge   $s0,1,endif # if argc < 1
      li    $v0,-1
      j     fim         # return -1;
endif:

while: # while ( (i < argc) && (i < 10) )
      bge   $s3,$s0,endwhile
      bge   $s3,10, endwhile
      sll   $t0,$s3,2   # t0=i*4;
      addu  $t0,$s1,$t0 # t0 = &argv[i];
      addi  $s3,$s3,1   # i++;
      lw    $a0,0($t0)  # a0 = argv[i++];
      li    $a1,10
      jal   atoi_t
      mtc1  $v0,$f4
      cvt.s.w $f4,$f4
            #(float) atoi_t(argv[i++],10)
      s.s   $f4,0($s2)  # *p = ...;
      addu  $s2,$s2,4   # p++;
      j while
endwhile:
      la    $a0,array
      move  $a1, $s0
      jal   mean
      mov.s $f12,$f0
      li    $v0,2
      syscall
        # print_float(mean(array, argc));
      li    $v0,0       # return 0;
fim:  lw    $s0,0($sp)
      lw    $s1,4($sp)
      lw    $s2,8($sp)
      lw    $s3,12($sp)
      lw    $ra,16($sp)
      addu  $sp,$sp,20
      jr    $ra
```

Nº Mec.: _____ Nome: _____ PROPOSTA DE RESOLUÇÃO _____ Turma: _____

2) Codifique em *assembly* do MIPS as seguintes funções `func1` e `func2`:

```
int func1(double *A, int p, int r)
{
    int i = p-1;
    int j = r+1;
    double piv = A[p];
    double tmp;
    for( ;; )
    {
       if(piv <= A[++i])
          j--;
       if ( i >= j )
          break;
       //swap
       tmp  = A[i];
       A[i] = A[j];
       A[j] = tmp;
    }
    return j;
}
void func2(double *A, int p, int r)
{
    int q;
    if(p < r)
    {
        q = func1(A, p, r);
        func2(A, p, q);
        func2(A, q+1, r);
    }
}
```

Label	Instrução em *assembly*	Comentário em C

```
        .text
# int func1(double *A, int p, int r){
# A -> a0; p -> a1; r -> a2
# i -> t1; t2 -> j; 8*i -> t3; 8*j -> t4
# piv -> f4; tmp -> f6
#
func1: addi $t1, $a1, -1  # t1 = i = p-1
       addi $t2, $a2, 1   # t2 = j = r+1
       #
       sll  $t0, $a1,3    # t0 = 8*p
       addu $t0, $a0,$t0  # t0 = &A[p]
       l.d  $f4, 0($t0)   # f4 = A[p]=piv
       #

f1_lp:                    # for(;;){
       addi $t1, $t1, 1   # ++i
       sll  $t3, $t1,3    #
       addu $t3, $t3,$a0  # t3 = &A[++i]
       l.d  $f6, 0($t3)   # f6 = A[++i]

       c.le.d $f4, $f6
            # if(piv<=A[++i]) f1_brk_if
       bc1f f1_brk_if     #
       addi $t2, $t2, -1  # j--
       #
f1_brk_if:
       bge  $t1, $t2,f1_for_end
            # if (i >= j) f1_for_end
            # swap A[i] and A[j]
       #l.d $f6, 0($t3)# tmp = A[i]
       # feito anteriormente!
       sll  $t4, $t2,3
       addu $t4, $a0,$t4  # t4 = &A[j]
       l.d  $f8, 0($t4)   # f8 = A[j]
       #
       s.d  $f8, 0($t3)   # A[i] = A[j]
       s.d  $f6, 0($t4)   # A[j] = tmp
       j    f1_lp         #
f1_for_end:               # }end_for_loop

       move $v0, $t2      # v0 = j
       jr   $ra
f1_end:                   # } //end_func1

# void func2(double *A, int p, int r){
# A -> a0; p -> a1; r -> a2
#
func2: addiu $sp, $sp,-20 # stack space
       sw   $ra, 0($sp)   # save $ra
       sw   $s0, 4($sp)   # save $s0
       sw   $s1, 8($sp)   # save $s1
       sw   $s2, 12($sp)  # save $s2
       sw   $s3, 16($sp)  # save $s3
       #
       bge  $a1,$a2,f2_ex # if(p>=r)
                          #    f2_ex
       # save a0, a1, a2
       move $s0, $a0      # s0 = a0 = A
       move $s1, $a1      # s1 = a1 = p
       move $s2, $a2      # s2 = a2 = r
       jal  func1         # func1(A, p,
r)
       #
       move $s3, $v0      # s3 = q
       #
       move $a0, $s0      # a0 = A
       move $a1, $s1      # a1 = p
       move $a2, $s3      # a2 = q
       jal  func2         # func2(A,p,q)
       #
       move $a0, $s0      # a0 = A
       addi $a1, $s3,1    # a1 = q+1
       move $a2, $s2      # a2 = r
       jal  func2         # func2(A,q+1,r)
       #
f2_ex: lw   $ra, 0($sp)   # restore $ra
       lw   $s0, 4($sp)   # restore $s0
       lw   $s1, 8($sp)   # restore $s1
       lw   $s2, 12($sp)  # restore $s2
       lw   $s3, 16($sp)  # restore $s3
       addiu $sp, $sp,20  # restore $sp
       #
       jr    $ra
f2_end:                   # } //end_func2
```


Ano Lectivo 2013/14 - 1º Semestre

Mini-Teste Prático 2 – Quinta-Feira, 12/12/2013, 9:00-11:00

Nº Mec.: _____ Nome: _____PROPOSTA DE RESOLUÇÃO _____ Turma: _____

NOTE BEM: Leia atentamente todas as questões, comente o código usando a linguagem C e respeite a convenção de passagem de parâmetros e salvaguarda de registos que estudou. Na tradução para o *Assembly* do MIPS respeite rigorosamente os aspectos estruturais e a sequência de instruções indicadas no código original fornecido.
O código em C apresentado pode não estar funcionalmente correcto, pelo que **não deve ser interpretado**.

1) Codifique em *assembly* do MIPS as seguintes funções `sqrt` e `func1`:

```
double absd(double);
double sqrt(double N, double tol)
{
    double oldguess = -1.0;
    double guess = 1.0;
    while( absd(guess-oldguess) > tol )
    {
        oldguess = guess;
        guess = (guess + N/guess) / 2;
    }
    return guess;
}

double func1(int a, int b, float c)
{
    double result;
    if(c == 0.0)
        result = func1(b, a, c);
    else
        result = (double)a;
    return sqrt(result, 0.1);
}
```

Label	Instrução em *assembly*	Comentário em C

```
        .data
dois:   .double     2.0
um:     .double     1.0
m_um:   .double -1.0
z_p_um: .double   0.1

        .text
sqrt:   # N   -> $f12; $f24
        # tol -> $f14; $f26
        # oldguess  -> $f20
        # guess         -> $f22

        subu  $sp,$sp,36
        sw    $ra,0($sp)
        s.d   $f20,4($sp)
        s.d   $f22,12($sp)
        s.d   $f24,20($sp)
        s.d   $f26,28($sp)

        mov.d $f24,$f12   # f24 = N;
        mov.d $f26,$f14   # f26 = tol;

        l.d   $f20,m_um   # oldguess = -1.0;
        l.d   $f22, um    # guess      = 1.0;
while:  sub.d $f12,$f22,$f20
        jal absd        #f0=absd(guess-oldguess);
        c.le.d $f0, $f26
        bc1t endwhile
        mov.d $f20,$f22   # oldguess = guess;
        div.d $f4, $f24,$f22    #f4=N/guess
        add.d $f4,$f4,$f22 #f4=guess+N/guess
        l.d   $f6, dois
        div.d $f22, $f4,$f6 #guess= f4/2;
        j     while
endwhile:
        mov.d $f0,$f22

        lw    $ra,0($sp)
        l.d   $f20,4($sp)
        l.d   $f22,12($sp)
        l.d   $f24,20($sp)
        l.d   $f26,28($sp)
        addu  $sp,$sp,36
        jr    $ra

func1:# a  ->  $a0
        # b  ->  $a1
        # c  ->  $f12

        subu  $sp,$sp,4
        sw    $ra,0($sp)

        mtc1  $0,$f0      #f0 = 0.0;
if:     c.eq.s $f12,$f0
        bc1f  else        # if (c == 0.0)
        move  $t0,$a0,
        move  $a0,$a1
        move  $a1,$t0
        jal   func1
              # result = func1(b, a, c);
        j endif
else:   mtc1  $a0,$f0
        cvt.d.w $f0,$f0   #result=(double) a;
endif:  mov.d $f12,$f0
        l.d   $f14,z_p_um
        jal sqrt
              #return sqrt(result,0.1)

        lw    $ra,0($sp)
        addu  $sp,$sp,4
        jr    $ra
```


Nº Mec.: _____ Nome: _____ PROPOSTA DE RESOLUÇÃO _____ Turma: _____

2) Codifique em *assembly* do MIPS a seguinte função `func2`:

```
#define SIZE 14
void func2(double *A,int lo, int mi, int hi)
{
    static double tmp[SIZE];
    int i = lo;
    int j = mi+1;
    int k = 0;

    while( i <= mi && j <= hi )
    {
        if( A[i] <= A[j] )
           tmp[k++] = A[i++];
        else
           tmp[k++] = A[j++];
    }
    i = lo;
    k = 0;
    while( i <= hi )
    {
       A[i++] = tmp[k++];
    }
}
```

Label	Instrução em *assembly*	Comentário em C

```
# void func2(double* A, int lo, int mi, int hi
){
# A -> a0; lo -> a1; mi -> a2; hi -> a3;
# I -> t0; j -> t1; k -> t2
# &A[i] -> t3; &A[j] -> t4;
# &tmp[k] -> t5; &tmp[0]=tmp -> t6
# A[i] -> f4; A[j] -> f6
#
        .data
tmp:    .space 112          # SIZE*8=14*8=112

        .text
func2:
       move   $t0, $a1      # i = lo
       addi   $t1, $a2, 1   # j = mi+1
       li     $t2, 0        # k =0
       #
       la     $t6, tmp      # t6 = tmp
f2_wh1:              # while(i<=mi && j<=hi) {
       bgt    $t0, $a2, f2_end_wh1
       bgt    $t1, $a3, f2_end_wh1
f2_if1:
       sll    $t3, $t0,3    #
       add    $t3, $t3,$a0
       l.d    $f4, 0($t3)   # f4 = A[i]

       sll    $t4, $t1,3    #
       add    $t4, $t4,$a0
       l.d    $f6, 0($t4)   # f6 = A[j]

       c.le.d $f4, $f6 #if(A[i]<=A[j]) f2_then1
       bc1f   f2_else1
#f2_then1:    # tmp[k++] = A[i++]
       sll    $t5, $t2,3    #
       add    $t5, $t5,$t6  # t5 = tmp + 8*k
       s.d    $f4,0($t5)    # tmp[k] = A[i];
       addi   $t0, $t0, 1   # i++
       j      f2_end_if1
f2_else1:                   # tmp[k++] = A[j++]
       sll    $t5, $t2,3    #
       add    $t5, $t5, $t6# t5 = tmp + 8*k
       s.d    $f6,0($t5)    # tmp[k] = A[j];
       addi   $t1, $t1, 1   # j++
f2_end_if1:
       addi   $t2, $t2, 1   # k++
       j      f2_wh1
f2_end_wh1:                 # } //end_wh1

       #
       move   $t0, $a1      # i=lo
       li     $t2, 0        # k=0
f2_wh2:              # while ( i <= hi ){
       bgt    $t0, $a3, f2_end_wh2
       #
       sll    $t5, $t2, 3   # 8*k
       add    $t5, $t5, $t6#
       l.d    $f4, 0($t5)   #f4 = tmp[k] (reuse f4!)
       #
       sll    $t3, $t0, 3   # 8*i
       add    $t3, $t3, $a0
       s.d    $f4, 0($t3)   # A[i]=tmp[k]
       #
       addi   $t2, $t2, 1   # k++
       addi   $t0, $t0, 1   # i++
       j      f2_wh2
f2_end_wh2:                 # } //end_wh2
       jr     $ra
f2_end:                     # } //end_func2
```