1η Άσκηση στην Αρχιτεκτονική Υπολογιστών
Assembly MIPS

Ανδρέας Στάμος
Αριθμός Μητρώου: *03120****

Νοέμβριος 2022

Περιεχόμενα

1 Μέρος Α

Ο συμπληρωμένος κώδικας είναι ο εξής:

```
addi $s1, $s0, 12
LOOP:
lw $t0, 0($s1)
beq $t0, $zero, END
div $t0, $s2
slti $t1, $t0, 50
beq $t1, $zero, ELSE
mfhi $t0
j NEXT
ELSE:
mflo $t0
NEXT:
sw $t0, 0($s1)
addi $s1, $s1, 4
j LOOP
END:

```

2 Μέρος Β

Προκειμένου να μειωθούν οι instructions έγιναν μερικές βελτιστοποιήσεις.

Αρχικά, αντί να γίνεται χρήση μεταβλητών index για να εντοπίζονται στοιχεία σε πίνακες, διατηρούμε σε μεταβλητές απευθείας τους αντίστοιχους δείκτες, δηλαδή αντικαθιστούμε την μεταβλητή i με την $A + 4i$ (το επί 4 διότι κάθε αριθμός καταλαμβάνει 1 λέξη, δηλαδή 4 θέσεις μνήμης).

Γενικότερα, αν μέσα σε εναν βρόχο γίνεται χρήση της μεταβλητής $f(i)$ και όχι της i, αντικαθιστούμε την μεταβλητή του βρόχου με την $i' = f(i)$ (προσαρμόζοντας ανάλογα το step και το exit condition).

Επιπλέον, στον βρόχο

```
for (i=0; i<n; ++i)
```

η συνθήκη $i < n$ αντικαθίσταται με την $i \neq n$ προκειμένου να αποφύγουμε την σύγκριση που θα κόστιζε μια επιπλέον instruction.

Τέλος, κάνουμε inline την συνάρτησης swap, κυρίως για να αποφύγουμε την διπλή ανάγνωση από την μνήμη των τιμών $A[i]$ και $A[i-1]$.

Αξίζει να επισημανθεί, πως στην υλοποίηση δεν έχει ληφθεί υπόψη το pipelining καθώς δεν είχε διδαχθεί ακόμα.

2.1 Bubble Sort

```
1  .data
2  #EXAMPLE
3  n: .word 6
4  example: .word 5, 7, 1, -1, -2, -1000
5
6  .text
7
8  .globl main
9
10 bubbleSort:
11 #$t0 := &A[N-i-1] (therefore the limits of the loop are from
   ↪  A+4*n-4 to A and the loop step is -4)
12
13 #&A[N-i-1] = &A[n-0-1]     (i=0)
14 addi $t0, $a1, -1
15 sll $t0, $t0, 2
16 add $t0, $t0, $a0
17
18 #$t1 NOT USED
19
20 #$t2 := swapped
21
22 L1:
23 #if !(&A[N-i-1]>&A): exit L1     (if i < N-1)
24 slt $t3, $a0, $t0
25 beq $t3, $zero, exitL1
26
27 or $t2, $zero, $zero #swapped = false
```

```
28
29  #$t3 := &A[j]
30  or $t3, $zero, $a0 #&A[j] = $A[0]
31
32  #$t4 NOT USED
33
34  L2:
35  #if !(&A[j] < &A[N-1-i]): exit L2
36  slt $t5, $t3, $t0
37  beq $t5, $zero, exitL2
38
39  lw $t5, 0($t3)
40  lw $t6, 4($t3)
41
42  #if !(A[j+1] < A[j]): do not swap
43  slt $t7, $t6, $t5
44  beq $t7, $zero, skipSwap
45
46  #A[j+1], A[j] = A[j], A[j+1]
47  sw $t5, 4($t3)
48  sw $t6, 0($t3)
49
50  ori $t2, $zero, 1 #swapped = true
51
52  skipSwap:
53  addi $t3, $t3, 4 #j++
54  j L2
55
56  exitL2:
57  beq $t2, $zero, exitL1
58  addi, $t0, $t0, -4 #i++ (reminder: $t0 = &A[n-i-1])
59  j L1
60
61  exitL1:
62  jr $ra
63
64
65  #ONLY FOR DEMONSTRATION PURPOSES
66  main:
67  la $a0, example
68  addi $sp, $sp, -4
69  sw $ra, 4($sp)
70  lw $a1, n
71  jal bubbleSort
72  lw $ra, 4($sp)
73  addi $sp, $sp, 4
74  jr $ra
75
```

2.2 Cocktail Sort

```
.data
#EXAMPLE
n: .word 6
example: .word 5, 7, 1, -1, -2, -1000

.text

.globl main

cocktailSort:
ori $t0, $zero, 1 #swapped = true
or $t1, $zero, $a0 #start = &A[0]

#end = &A[n-1]
or $t2, $zero, $a1
addi $t2, $t2, -1
sll $t2, $t2, 2
add $t2, $t2, $a0

L1:
beq $t0, $zero, exitL1
or $t0, $zero, $zero #swapped = false
or $t3, $zero, $t1 #i = &A[start]

L2:
#if !(i<end): exitL2
slt $t4, $t3, $t2
beq $t4, $zero, exitL2

lw $t4, 0($t3) #$t4 = A[i]
lw $t5, 4($t3) #$t5 = A[i+1]

#if !(A[i+1]<A[i]) skip the swap
slt $t6, $t5, $t4
beq $t6, $zero, skipSwapL2

#A[i], A[i+1] = A[i+1], A[i]
sw $t4, 4($t3)
sw $t5, 0($t3)

ori $t0, $zero, 1 #swapped = true

skipSwapL2:
addi $t3, $t3, 4 #i++
j L2

exitL2:
beq $t0, $zero, exitL1
```

```
49  or $t0, $zero, $zero #swapped = false
50  addi $t2, $t2, -4 #end--
51  addi $t3, $t2, -4 #i = &A[end-1]
52
53  L3:
54  #if i<start: exitL3
55  slt $t4, $t3, $t1
56  bne $t4, $zero, exitL3
57
58  lw $t4, 0($t3) #$t4 = A[i]
59  lw $t5, 4($t3) #$t4 = A[i+1]
60
61  #if !(A[i+1]<A[i]) skip the swap
62  slt $t6, $t5, $t4
63  beq $t6, $zero, skipSwapL3
64
65  #A[i], A[i+1] = A[i+1], A[i]
66  sw $t4, 4($t3)
67  sw $t5, 0($t3)
68
69  ori $t0, $zero, 1 #swapped = true
70
71  skipSwapL3:
72  addi $t3, $t3, -4 #i--
73  j L3
74
75  exitL3:
76  addi $t1, $t1, 4 #start++
77  j L1
78
79  exitL1:
80  jr $ra
81
82
83  #ONLY FOR DEMONSTRATION PURPOSES
84  main:
85  la $a0, example
86  addi $sp, $sp, -4
87  sw $ra, 4($sp)
88  lw $a1, n
89  jal cocktailSort
90  lw $ra, 4($sp)
91  addi $sp, $sp, 4
92  jr $ra
93
```

2.3 Insertion Sort (iterative)

```
1  .data
2  #EXAMPLE
```

```
n: .word 6
example: .word 5, 7, 1, -1, -2, -1000

.text

.globl main

insertionSort:

#$t0 = &A[N]
or $t0, $zero, $a1
sll $t0, $t0, 2
add $t0, $t0, $a0

addi $t1, $a0, 4 #i = &A[1]

L1:
#if i == n: exit L1
beq $t1, $t0, exitL1

lw $t2, 0($t1) #key = A[i]

or $t3, $zero, $t1 #$t3 = &A[j+1] = $A[i]

L2:
#if &A[j+1] == &A: exitL2   (equivalantly if j == -1)
beq $t3, $a0, exitL2

lw $t4, -4($t3)  #key = &A[j] = &A[j+1] - 4

#if !(key<A[j]): exitL2
slt $t5, $t2, $t4
beq $t5, $zero, exitL2

sw $t4, 0($t3) #A[j+1] = A[j]
addi $t3, $t3, -4 #j--
j L2 #loop L2

exitL2:
sw $t2, 0($t3) #A[j+1] = key
addi $t1, $t1, 4 #i++
j L1 #loop L1

exitL1:
jr $ra


#ONLY FOR DEMONSTRATION PURPOSES
main:
la $a0, example
```

```
addi $sp, $sp, -4
sw $ra, 4($sp)
lw $a1, n
jal insertionSort
lw $ra, 4($sp)
addi $sp, $sp, 4
jr $ra

```

3 Μέρος Γ

Ο αναδρομικός αλγόριθμος φαίνεται στον παρακάτω κώδικα C:

```
void insertionSort(int *const A, int N) {
        --N; //N is now the index of last element
        if (N == 0) return;
        insertionSort(A, N); //N-1 elements

        int *j = A+N;
        const int key = *j;
        while (j != A && *(j-1) > key) {
                *j = *(j-1);
                --j;
        }
        *j = key;
}

```

Ο αλγόριθμος υλοποίεται σε MIPS Assembly ως εξής:

```
.data
#EXAMPLE
n: .word 6
example: .word 5, 7, 1, -1, -2, -1000

.text

.globl main

insertionSort_rec:
addi $a1, $a1, -1

#if N==0: return
beq $a1, $zero, exitFunc

addi $sp, $sp, -12
sw $a0, 4($sp)
sw $a1, 8($sp)
sw $ra, 12($sp)

```

```
21  jal insertionSort_rec
22
23  lw $a0, 4($sp)
24  lw $a1, 8($sp)
25  lw $ra, 12($sp)
26  addi $sp, $sp, 12
27
28  #j = A+N
29  sll $a1, $a1, 2
30  add $t0, $a0, $a1
31
32  lw $t1, 0($t0) #key = *j
33
34  L1:
35  beq $t0, $a0, exitL1 #if j == A: exit L1
36
37  #if !(key < *(j-1)): exit L1
38  lw $t2, -4($t0)
39  slt $t3, $t1, $t2
40  beq $t3, $zero, exitL1
41
42  sw $t2, 0($t0) #*j = *(j-1)
43  addi $t0, $t0, -4 #--j;
44
45  j L1
46
47  exitL1:
48  sw $t1, 0($t0)
49
50  exitFunc:
51  jr $ra
52
53
54  #ONLY FOR DEMONSTRATION PURPOSES
55  main:
56  la $a0, example
57  addi $sp, $sp, -4
58  sw $ra, 4($sp)
59  lw $a1, n
60  jal insertionSort_rec
61  lw $ra, 4($sp)
62  addi $sp, $sp, 4
63  jr $ra
64
```