182.690 RECHNERSTRUKTUREN – INSTRUCTIONS (3)

Thomas Polzer
tpolzer@ecs.tuwien.ac.at
Institut für Technische Informatik

C-Sort Example

- Can be found in textbook (pages 149 – 155)
- Big example to wrap-up the instruction chapter
- Two C-Functions
  - swap
  - sort

Swap Function

- C-code

```
void swap(int v[], int k)
{
        int temp;
        temp = v[k];
        v[k] = v[k + 1];
        v[k + 1] = temp;
}
```

Swap Function – Register Allocation

Register	C-Name	Type	Description
$a0	v	Parameter	Base address of the array
$a1	k	Parameter	Array index
$t0	temp	Local variable	Temporary storage for v[k]
$t1			Calculated address of v[k]
$t2			Temporary storage for v[k+1]

Swap Function – Assembler Code

```
swap:   sll $t1, $a1, 2      Calculate address of v[k]
        add $t1, $a0, $t1
        lw $t0, 0($t1)       temp = v[k]
        lw $t2, 4($t1)       v[k] = v[k + 1]
        sw $t2, 0($t1)
        sw $t0, 4($t1)       v[k + 1] = temp
        jr $ra               Return
```

Swap Function - Remarks

- Return address and parameters are not saved to stack (Leaf procedure)
- Two additional registers required ($t1 and $t2)
- Temporary registered not preserved!

Sort Function

- C-code

```
void sort(int v[], int n)
{
  int i, j;
  for (i=0; i<n; i++)
  {
    for (j=i-1; j>=0 && v[j]>v[j+1]; j--)
      swap(v, j);
  }
}
```

Sort Function – Register Allocation

Register	C-Name	Type	Description
$a0, $s2	v	Parameter	Base address of the array
$a1, $s3	n	Parameter	Array size
$s0	i	Local variable	Index of outer loop
$s1	j	Local variable	Index of inner loop
$t0			Used for loop condition calculation
$t1			Intermediate address
$t2			Address of v[j]
$t3			v[j]
$t4			v[j+1]

Sort Function – Outer Loop

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

```
            move $s0, $zero          i=0
for1tst:    slt $t0, $s0, $a1        i<n
            beq $t0, $zero, exit1    If not, stop the loop
            …                        Loop body
            addi $s0, $s0, 1         i++
            j for1tst                Repeat with next
                                     iteration
exit1:
```

Sort Function – Inner Loop

```
for (j=i-1; j>=0 && v[j]>v[j+1]; j--)
            addi $s1, $s0, -1       j=i-1
for2tst:    slti $t0, $s1, 0        s1<0
            bne $t0, $zero, exit2   If not stop the loop
            sll $t1, $s1, 2         Calculate the address of
            add $t2, $a0, $t1       v[j]
            lw $t3, 0($t2)          Load v[j]
            lw $t4, 4($t2)          Load v[j+1]
            slt $t0, $t4, $t3       v[j+1] < v[j]
            beq $t0, $zero, exit2   If so stop the loop
            …                       Loop body
            addi $s1, $s1, -1       j--
            j for2tst               Next iteration
exit2:
```

Sort Function – Calling swap

```
swap(v, j)
```

```
jal swap
```

Sort Function - Putting it all Together

- We now have all required parts
- But two problems remain
  - Swap overwrites the return address of sort and does not have parameters set
  - Sort does not restore the save registers!

Sort Function – Saving the Parameters

- At the beginning of sort:

```
move $s2, $a0
move $s3, $a1
```

- Replace all occurrences of
  - $a0 with $s2 and
  - $a1 with $s3
- Before calling swap:

```
move $a0, $s2     Passing v
move $a1, $s1     Passing j
```

Sort Function – Preserving Registers

At the beginning of sort:

```
addi $sp, $sp, -20
sw $ra, 16($sp)
sw $s3, 12($sp)
sw $s2, 8($sp)
sw $s1, 4($sp)
sw $s0, 0($sp)
```

At the end of sort:

```
lw $s0, 0($sp)
lw $s1, 4($sp)
lw $s2, 8($sp)
lw $s3, 12($sp)
lw $ra, 16($sp)
addi $sp, $sp, 20
```

Sort Function - Putting it all Together

```
sort:   addi $sp, $sp, -20
        sw $ra, 16($sp)
        sw $s3, 12($sp)          Save registers
        sw $s2, 8($sp)
        sw $s1, 4($sp)
        sw $s0, 0($sp)
        move $s0, $zero          i=0
for1tst: slt $t0, $s0, $a1       i<n
        beq $t0, $zero, exit1    If not, stop the loop
        addi $s1, $s0, -1        j=i-1
for2tst: slti $t0, $s1, 0        s1<0
        bne $t0, $zero, exit2    If not stop the loop
        sll $t1, $s1, 2          Calculate the
        add $t2, $a0, $t1        address of v[j]
        lw $t3, 0($t2)           Load v[j]
        lw $t4, 4($t2)           Load v[j+1]
        slt $t0, $t4, $t3        v[j+1] < v[j]
        beq $t0, $zero, exit2    If so stop the loop
        move $a0, $s2            Passing v
        move $a1, $s1            Passing j
        jal swap
        addi $s1, $s1, -1        j--
        j for2tst                Next iteration
exit2:  addi $s0, $s0, 1         i++
        j for1tst                Repeat with next
                                 iteration
exit1:  lw $s0, 0($sp)
        lw $s1, 4($sp)
        lw $s2, 8($sp)           Restore registers
        lw $s3, 12($sp)
        lw $ra, 16($sp)
        addi $sp, $sp, 20
        jr $ra                   Return
```

Machine Code

```
mips-elf-as -mips1 -o test test.s
mips-elf-objdump -d test

Disassembly of section .text:

00000000 <sort>:
   0:   23bdffec        addi    sp,sp,-20
   4:   afbf0010        sw      ra,16(sp)
   8:   afb3000c        sw      s3,12(sp)
   c:   afb20008        sw      s2,8(sp)
  10:   afb10004        sw      s1,4(sp)
  14:   afb00000        sw      s0,0(sp)
  18:   00008021        move    s0,zero

0000001c <for1tst>:
  1c:   0205402a        slt     t0,s0,a1
  20:   11000014        beqz    t0,74 <exit1>
  24:   00000000        nop
  28:   2211ffff        addi    s1,s0,-1
```

Required for correct execution

We will discuss the problem later when looking at pipelining

Machine Code

```
0000002c <for2tst>:
  2c:   2a280000        slti    t0,s1,0
  30:   1500000e        bnez    t0,6c <exit2>
  34:   00000000        nop
  38:   00114880        sll     t1,s1,0x2
  3c:   00895020        add     t2,a0,t1
  40:   8d4b0000        lw      t3,0(t2)
  44:   8d4c0004        lw      t4,4(t2)
  48:   00000000        nop
  4c:   018b402a        slt     t0,t4,t3
  50:   11000006        beqz    t0,6c <exit2>
  54:   00000000        nop
  58:   02402021        move    a0,s2
  5c:   0c000025        jal     94 <swap>
  60:   02202821        move    a1,s1
  64:   0800000b        j       2c <for2tst>
  68:   2231ffff        addi    s1,s1,-1

0000006c <exit2>:
  6c:   08000007        j       1c <for1tst>
  70:   22100001        addi    s0,s0,1
```

Machine Code

```
00000074 <exit1>:
  74:   8fb00000        lw      s0,0(sp)
  78:   8fb10004        lw      s1,4(sp)
  7c:   8fb20008        lw      s2,8(sp)
  80:   8fb3000c        lw      s3,12(sp)
  84:   8fbf0010        lw      ra,16(sp)
  88:   23bd0014        addi    sp,sp,20
  8c:   03e00008        jr      ra
  90:   00000000        nop

00000094 <swap>:
  94:   00054880        sll     t1,a1,0x2
  98:   00894820        add     t1,a0,t1
  9c:   8d280000        lw      t0,0(t1)
  a0:   8d2a0004        lw      t2,4(t1)
  a4:   00000000        nop
  a8:   ad2a0000        sw      t2,0(t1)
  ac:   03e00008        jr      ra
  b0:   ad280004        sw      t0,4(t1)
```

182.690 RECHNERSTRUKTUREN – INSTRUCTIONS (3)

Thomas Polzer
tpolzer@ecs.tuwien.ac.at
Institut für Technische Informatik