Computer Systems Engineering

Computer Architecture

Homework #1

Student:
Ameer Alkam
ID:
1120217

Supervisor:
Dr. Aziz Qaroush

March 9, 2016

1 Excercise 2.4

1.1 From the following C code:

a	$f = -g - A[4];$
b	$B[8] = A[i - j];$

1.1.1 MIPS instructions corresponding to it

First statement:

- sub $t0, $zero, $s1
- lw $t1, 16($s6)
- sub $t1, $zero, $t1
- add $s0, $t0, $t1

Second statement:

- sub $t0, $s3, $s4
- sll $t0, $t0, 4
- add $t0, $t0, $s6
- lw $t1, ($t0)
- sw $t1, 32($s7)

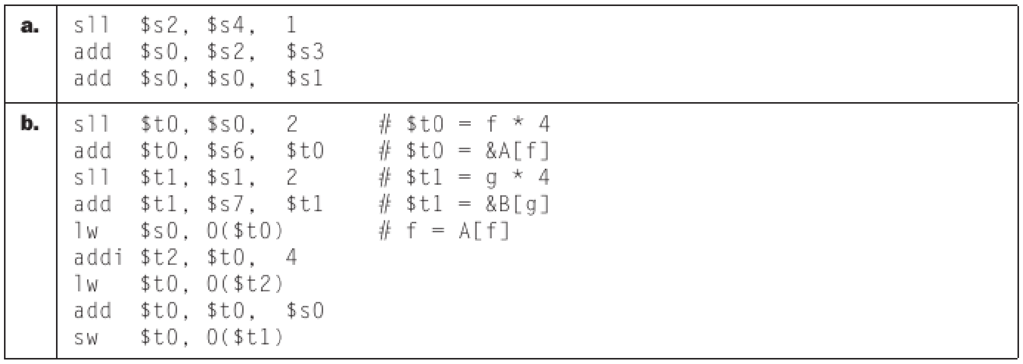
1.1.2 For the C statements how many MIPS instructions were needed?

4 MIPS instruction were needed for first statement, while 5 where required by the second.

1.1.3 How many different registers needed to carry out the instructions?

7 registers that held the original variables. As well as two temporary registers needed to properly take data from memory to add/sub.

1.2 From the following MIPS code:



a.

```
sll  $s2, $s4,  1
add  $s0, $s2,  $s3
add  $s0, $s0,  $s1
```

b.

```
sll  $t0, $s0,  2      # $t0 = f * 4
add  $t0, $s6,  $t0    # $t0 = &A[f]
sll  $t1, $s1,  2      # $t1 = g * 4
add  $t1, $s7,  $t1    # $t1 = &B[g]
lw   $s0, 0($t0)       # f = A[f]
addi $t2, $t0,  4
lw   $t0, 0($t2)
add  $t0, $t0,  $s0
sw   $t0, 0($t1)
```

1.2.1 C code corresponding to it:

First set of instructions:

- f = ((j<<1) + i) + g;

Second set of instructions:

- B[g] = A[f]+A[f + 4];
- f = A[f];

1.2.2 Minimize the number of instructions needed.

The instructions given can't be further minimized.

1.2.3 How many registers are needed?

First set of instructions some

Second set of instructions other

2 Excercise 2.6

2.1 From the following C code:

a.	`f = f + A[2];`
b.	`B[8] = A[i] + A[j];`

2.1.1 MIPS instructions corresponding to it:

First statement:

- lw \$t0, 4(\$s6)
- add \$s0, \$s0, \$t0

Second statement:

- add \$t1, \$s3, \$s6
- sll \$t1, \$t1, 2
- lw \$t1, (\$t1)
- add \$t2, \$s4, \$s6
- sll \$t2, \$t2, 2
- lw, \$t2, (\$t2)
- add \$t0, \$t1, \$t2
- sw \$t0, 32(\$s7)

2.1.2 For the C statements how many MIPS instructions were needed?

2 MIPS instruction were needed for each of the first C statements, while the second required 8 instructions.

2.1.3 How many registers were needed?

7 registers that held the original variables. As well as three temporary registers needed to properly take data from memory to add/sub.

2.2 From the following MIPS code:

a.

```
sub $s0, $s0, $s1
sub $s0, $s0, $s3
add $s0, $s0, $s1
```

b.

```
addi $t0, $s6, 4
add  $t1, $s6, $0
sw   $t1, 0($t0)
lw   $t0, 0($t0)
add  $s0, $t1, $t0
```

2.2.1 C code corresponding to it:

First set of instructions:

- f -= g;
- f -= h;
- f += g;

Second set of instructions:

- A[4] = &A[0];
- f = &A[0] + A[4];

2.2.2 Value at $s0 at the end of assembly code:

First set of instructions:

- $s0 = 0xffffffe2

Second set of instructions:

- $s0 = 0x00000200

2.2.3 Opcode and register values:

Intruction	Opcode_Rs_Rt_{Rd or Immediate}_{func}
sub $s0, $s0, $s1	000000_10000_100010_10000_100010
sub $s0, $s0, $s3	000000_10000_10011_10000_100010
add $s0, $s0, $s1	000000_10000_100010_10000_100000
addi $t0, $s6, 4	001000_10110_01001_0000000000000100
add $t1, $s6, $0	000000_10110_00000_01001_100000
sw $t1, 0($t0)	101011_01001_01001_0000000000000000
lw $t0, 0($t0)	100011_01001_01001_0000000000000000
add $s0, $t1, $t0	000000_01001_01000_10000_100000

3 Excercise 2.10

4 Excercise 2.12

4.1 Given these changes to MIPS ISA:

a.	128 registers
b.	Four times as many different instructions

4.1.1 For each of the changes, the formatting of the R-instructions would be as follows.

For the first change:

- Using 128 registers would mean that the 5 bits used to address the registers won't be enough, now each register must have 7 bits to be identified.
  - If instruction length is to remain 32 bits the formatting would become:
  -

Opcode:6	Rs:7	Rt:7	Rs:7	Sh:0	Func:5

For the second change:

- Assuming the added instructions would affect only the Opcode, (no extra R-type instructions added)
  - Keeping the instruction length to 32 bits the formatting would be
  -

Opcode:10	Rs:5	Rt:5	Rs:5	Sh:1	Func:6

- Assuming the added instructions would affect only the functions, (all added instructions are R-type functions)
  - Keeping the instruction length to 32 bits the formatting would be
  -

Opcode:6	Rs:5	Rt:5	Rs:5	Sh:1	Func:10

5 Excercise 2.17

5.1 The following instructions aren't included in MIPS-ISA:

a.	`subi $t2, $t3, 5`	`# R[rt] = R[rs] - SignExtImm`
b.	`rpt  $t2, loop`	`# if(R[rs]>0) R[rs]=R[rs]-1, PC=PC+4+BranchAddr`

5.1.1 Why aren't they included in MIPS-ISA?

subi	isn't needed since the instruction 'addi' takes a signed immediate, so having subi would be redundant.
rpt	can be implemented simply with 'sub' or 'addi' along with branching instructions, which give the same effect as rpt without wasting instructions.

5.1.2 If they were implemented, what would be the most appropriate instruction format?

Both being of the Immediate instruction type.

5.1.3 Shortest MIPS instructions seuence to perform the same operation.

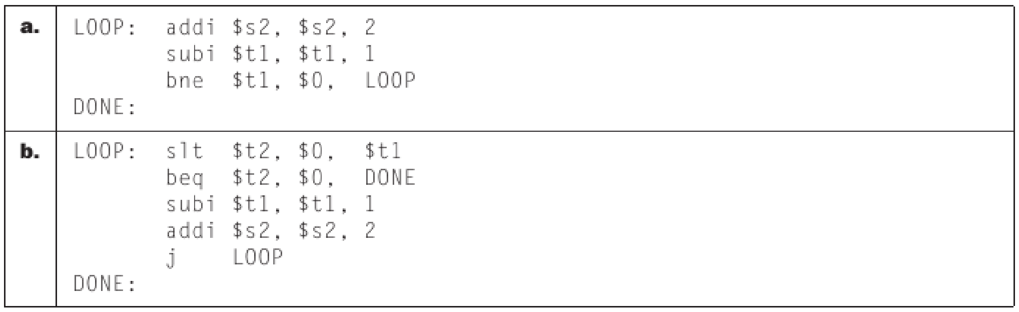
subi $t2, $t3, 5

- addi $t2, $t3, -5

rpt $t2, loop

- repeat:
  - # instructions here
  - addi $t2, $t2, -1
- bnq $t2, $zero, repeat

5.2 For the following instructions



a.	`LOOP:  addi $s2, $s2, 2`
	`       subi $t1, $t1, 1`
	`       bne  $t1, $0,  LOOP`
	`DONE:`

b.	`LOOP:  slt  $t2, $0,  $t1`
	`       beq  $t2, $0,  DONE`
	`       subi $t1, $t1, 1`
	`       addi $s2, $s2, 2`
	`       j    LOOP`
	`DONE:`

5.2.1 Assuming initially $t is 10, $s2 is 0, after executing the instructions, what would the value of $s be?

1. 20.
2. 20.

5.2.2 Equivilant C code.

a

- for(; i>0; i–){
    - B+=2;
- }

b

- while(i>0){
    - B+=2;
    - i–;
- }

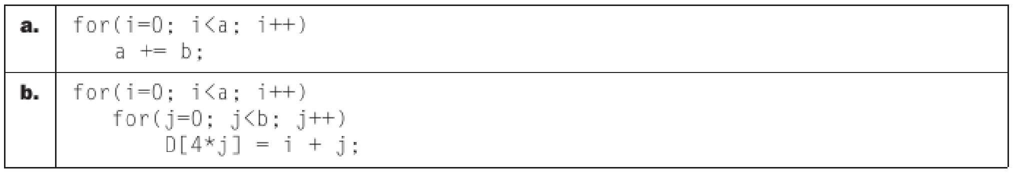
5.2.3 If $t1 initially had number N in it.

a 2N instructions would be executed.

b 5N + 2 instructions would be executed.

6 Excercise 2.18

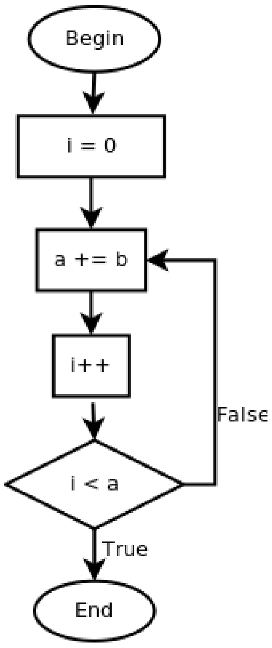
6.1 For the following C code.



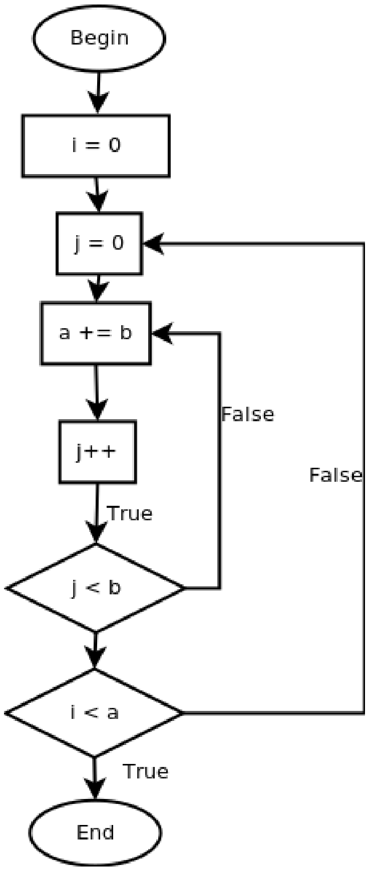
a.	`for(i=0; i<a; i++)`
	`    a += b;`

b.	`for(i=0; i<a; i++)`
	`    for(j=0; j<b; j++)`
	`        D[4*j] = i + j;`

6.1.1 Draw control flow graph.



(a)



(b)

6.1.2 The above code in MIPS assembly.

a

- addi \$t0, \$zero, 0
- loop:
  - add \$s0, \$s0, \$s1
  - addi \$t0, \$t0, 1
- slt \$t2, \$t0, \$s0
- bnq \$t2, \$zero, loop

b

- addi $t0, $zero, 0
- loop_outer:
  - addi $t1, $zero, 0
  - loop_inner:
    - sll $t2, $t1, 2
    - add $t2, $t2, $s2
    - add $t3, $t0, $t1
    - lw $t3, ($t2)
    - addi $t1, $t1, 1
  - slt $t2, $t1, $s1
  - bne $t2, $zero, loop_inner
  - addi $t0, $t0, 1
- slt $t2, $t0, $s0
- bne $t2, $zero, loop_outer

6.1.3 How many MIPS instructions were needed to implement the C code. Assuming a, b are initially 10 and 1, all elements of D are 0, how many MIPS instructions would be executed?

Total number of intructions to implement it

- The first loop: 5 instructions and one label.
- The second loop: 12 instructions and 2 labels.

Number of instructions executed given the previous initial vlaues

- The first loop: is an infinite loop, 'a' is being increased at the same rate as 'i'.
- The second loop: 111 instructions would be executed.

6.2 For the following MIPS code.

a.	`      addi $t1, $0, 50`
	`LOOP: lw   $s1, 0($s0)`
	`      add  $s2, $s2, $s1`
	`      lw   $s1, 4($s0)`
	`      add  $s2, $s2, $s1`
	`      addi $s0, $s0, 8`
	`      subi $t1, $t1, 1`
	`      bne  $t1, $0, LOOP`

b.	`      addi $t1, $0, $0`
	`LOOP: lw   $s1, 0($s0)`
	`      add  $s2, $s2, $s1`
	`      addi $s0, $s0, 4`
	`      addi $t1, $t1, 1`
	`      slti $t2, $t1, 100`
	`      bne  $t2, $s0, LOOP`

6.2.1 What's the total number of MIPS instructions being executed?

a 351 instructions.

b 601 instructions.

6.2.2 The above code in C.

a

- for(i=50; i>0; i–){
  - result += MemArray[0];
  - result += MemArray[1];
  - MemArray += 2;
- }

b

- for(i=0; i<100; i++){
  - result += MemArray[0];
  - MemArray += 1;
- }

6.2.3 The above code reduced would be.