CSE 31
Computer Organization

Lecture 16 – MIPS Conditionals (wrap up)

Announcements

- Labs
  - Lab 5 grace period ends this week
    - » No penalty for submission during grace period
    - » Demo is REQUIRED to receive full credit
  - Lab 6 due this week (with **14 days grace period** after due date)
    - » Demo is REQUIRED to receive full credit
  - Lab 7 out this week
    - » Due at 11:59pm on the same day of your lab after next (with 7 days grace period after due date)
    - » You must demo your submission to your TA within 21 days from posting of lab
    - » Demo is REQUIRED to receive full credit
- Reading assignments
  - Reading 04 (zyBooks 4.1 – 4.9) due **tonight**, 20-MAR and Reading 05 (zyBooks 1.6 – 1.7, 6.1 – 6.3) due 03-APR
    - » Complete **Participation Activities** in each section to receive grade
    - » IMPORTANT: Make sure to submit score to CatCourses by using the link provided on CatCourses

Announcements

- Homework assignment
  - Homework 04 (zyBooks 4.1 – 4.9) due 03-APR
    - Complete **Challenge Activities** in each section to receive grade
    - IMPORTANT: Make sure to submit score to CatCourses by using the link provided on CatCourses
- Project 02
  - Due 05-MAY
  - Can work in teams of 2 students
    - Each team member must identify teammate in "Comments…" text-box at the submission page
    - If working in teams, each student must submit code (can be the same as teammate) and demo individually
    - Grade can vary among teammates depending on demo
  - Demo required for project grade
    - No partial credit for submission without demo
  - **No grace period**
    - **Must complete submission and demo by due date.**

Inequalities in MIPS (1/4)

- Until now, we've only tested equalities (`==` and `!=` in C). General programs need to test **<** and **>** as well.

- Introduce MIPS Inequality Instruction:
  - "Set on Less Than"
  - Syntax: `slt reg1,reg2,reg3`
  - Meaning: reg1 = (reg2 < reg3);

```
if (reg2 < reg3)
     reg1 = 1;
else reg1 = 0;
```

Same thing…

"set" means "change to 1",
"reset" means "change to 0".

Inequalities in MIPS (2/4)

- How do we use this? Compile by hand:

```
if (g < h) goto Less; #g:$s0, h:$s1
```

- Answer: compiled MIPS code…

```
slt $t5,$s0,$s1 # $t5 = 1 if g < h
bne $t5,$0,Less # goto Less if $t5 != 0
                # (if (g < h)) Less:
```

Why not **beq $t5, 1, Less**?

- Register **`$0`** always contains the value `0`, so `bne` and `beq` often use it for comparison after a `slt` instruction.

- A **`slt`** → **`bne`** pair means **`if(… < …)goto…`**

Inequalities in MIPS (3/4)

- Now we can implement <, but how do we implement >, ≤ and ≥ ?

- We could add 3 more instructions, but:
  - MIPS goal: Simpler is Better

- Can we implement ≤ in one or more instructions using just `slt` and branches?
  - What about >?
  - What about ≥?

Inequalities in MIPS (4/4)

```
# a:$s0, b:$s1
slt $t0,$s0,$s1 # $t0 = 1 if a < b
beq $t0,$0,skip # skip if a >= b
    <stuff>     # do if a < b
skip:
```

How about **>** and **<=**?

Two independent variations possible:

Use `slt $t0,$s1,$s0` instead of `slt $t0,$s0,$s1`

Use `bne` instead of `beq`

Immediates in Inequalities

- There is also an immediate version of `slt` to test against constants: `slti`
  - Helpful in **`for`** loops

```
    if (g >= 1) goto Loop
```

C

MIPS

```
Loop:       . . .
 slti $t0,$s0,1        # $t0 = 1 if
                       # $s0 < 1 (g < 1)
 beq  $t0,$0,Loop      # goto Loop
                       # if $t0==0
                       # (if (g>=1))
```

An `slt` ➔ `beq` pair means `if(… ≥ …)goto…`

What about unsigned numbers?

- Also unsigned inequality instructions:

  `sltu`, `sltiu`

…which sets result to **1** or **0** depending on unsigned comparisons

- What is value of `$t0`, `$t1`?

- (`$s0 = FFFF FFFA`$_{hex}$, `$s1 = 0000 FFFA`$_{hex}$)

```
slt  $t0, $s0, $s1  1
sltu $t1, $s0, $s1  0
```

MIPS Signed vs. Unsigned – diff meanings!

- MIPS terms Signed/Unsigned “overloaded”:
  - Do/Don't sign extend
    - `(lb, lbu)`
  - Do/Don't overflow
    - `(add, addi, sub, mult, div)`
    - `(addu, addiu, subu, multu, divu)`
  - Do signed/unsigned compare
    - `(slt, slti / sltu, sltiu)`

Example: The C Switch Statement (1/3)

- Choose among four alternatives depending on whether `k` has the value 0, 1, 2 or 3. Compile this C code:

```
switch (k) {
 case 0: f=i+j; break; /* k=0 */
 case 1: f=g+h; break; /* k=1 */
 case 2: f=g-h; break; /* k=2 */
 case 3: f=i-j; break; /* k=3 */
}
```

Example: The C Switch Statement (2/3)

- This is complicated, so simplify.

- Rewrite it as a chain of if-else statements, which we already know how to compile:

```
if(k==0) f=i+j;
   else if(k==1) f=g+h;
      else if(k==2) f=g-h;
         else if(k==3) f=i-j;
```

- Use this mapping:

```
f:$s0, g:$s1, h:$s2,
i:$s3, j:$s4, k:$s5
```

Example: The C Switch Statement (3/3)

- Final compiled MIPS code:

```
      bne  $s5,$0,L1      # branch k!=0
      add  $s0,$s3,$s4 # k==0 so f=i+j
      j    Exit           # end of case so Exit
L1:   addi $t0,$s5,-1     # $t0=k-1
      bne  $t0,$0,L2      # branch k!=1
      add  $s0,$s1,$s2 # k==1 so f=g+h
      j    Exit           # end of case so Exit
L2:   addi $t0,$s5,-2     # $t0=k-2
      bne  $t0,$0,L3      # branch k!=2
      sub  $s0,$s1,$s2 # k==2 so f=g-h
      j    Exit           # end of case so Exit
L3:   addi $t0,$s5,-3     # $t0=k-3
      bne  $t0,$0,Exit # branch k!=3
      sub  $s0,$s3,$s4 # k==3 so f=i-j
Exit:
```

Always compared with $0!

Quiz

```
Loop:addi $s0,$s0,-1    # i = i - 1
     slti $t0,$s1,2     # $t0 = (j < 2)
     beq  $t0,$0 ,Loop # goto Loop if $t0 == 0
     slt  $t0,$s1,$s0   # $t0 = (j < i)
     bne  $t0,$0 ,Loop # goto Loop if $t0 != 0
```

($s0=i, $s1=j)

What C code properly fills in the blank in loop below?

```
do {i--;} while(__);
```

1) j < 2 && j < i
2) j ≥ 2 && j < i
3) j < 2 && j ≥ i
4) j ≥ 2 && j ≥ i
5) j > 2 && j < i
6) j < 2 || j < i
7) j ≥ 2 || j < i
8) j < 2 || j ≥ i
9) j ≥ 2 || j ≥ i
10) j > 2 || j < i

Quiz

```
Loop:addi $s0,$s0,-1     # i = i - 1
     slti $t0,$s1,2      # $t0 = (j < 2)
     beq  $t0,$0 ,Loop # goto Loop if $t0 == 0
     slt  $t0,$s1,$s0    # $t0 = (j < i)
     bne  $t0,$0 ,Loop # goto Loop if $t0 != 0
```

($s0=i, $s1=j)

What C code properly fills in the blank in loop below?

```
do {i--;} while(__);
```

1) j < 2 && j < i

2) j ≥ 2 && j < i

3) j < 2 && j ≥ i

4) j ≥ 2 && j ≥ i

5) j > 2 && j < i

6) j < 2 || j < i

7) j ≥ 2 || j < i

8) j < 2 || j ≥ i

9) j ≥ 2 || j ≥ i

10) j > 2 || j < i

Summary of MIPS Conditionals

- To help the conditional branches make decisions concerning inequalities, we introduce: “Set on Less Than” called **`slt`**, **`slti`**, **`sltu`**, **`sltiu`**

- One can store and load (signed and unsigned) bytes as well as words with `lb`, `lbu`

- Unsigned add/sub doesn’t cause overflow

- New MIPS Instructions:

```
    sll, srl, lb, lbu
    slt, slti, sltu, sltiu
    addu, addiu, subu
```