

Chapter 2

Instructions: Language of the Computer



Conditional Operations

- Branch to a labeled instruction if a condition is true
 - Otherwise, continue sequentially
- `beq rs, rt, L1`
 - if (`rs == rt`) branch to instruction labeled L1;
- `bne rs, rt, L1`
 - if (`rs != rt`) branch to instruction labeled L1;
- `j L1`
 - unconditional jump to instruction labeled L1



Compiling If Statements

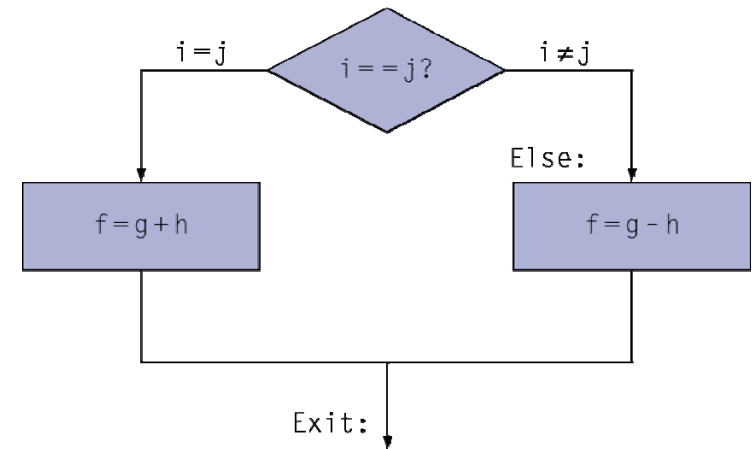
- C code:

```
if (i == j) f = g+h;  
else f = g-h;
```

- f, g, ... in \$s0, \$s1, ...

- Compiled MIPS code:

```
        bne $s3, $s4, Else  
        add $s0, $s1, $s2  
        j    Exit  
Else:   sub $s0, $s1, $s2  
Exit:   ...
```



Assembler calculates addresses



Compiling Loop Statements

- C code:

```
while (save[i] == k) i += 1;
```

- i in \$s3, k in \$s5, address of save in \$s6

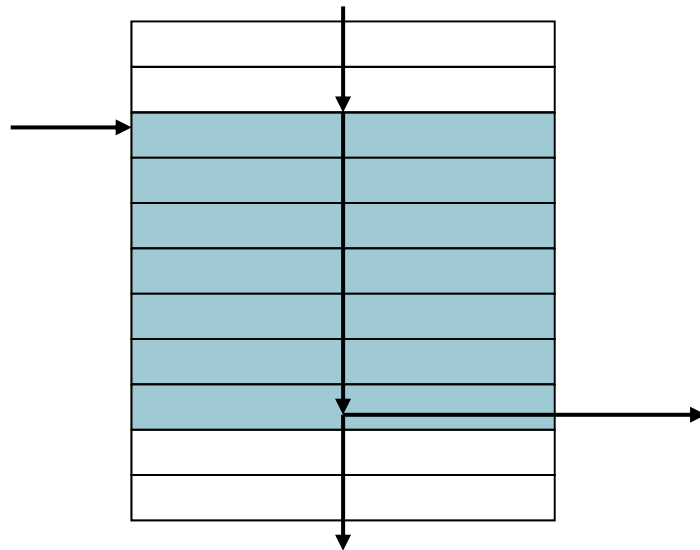
- Compiled MIPS code:

```
Loop:  slt    $t1, $s3, 2
        add   $t1, $t1, $s6
        lw    $t0, 0($t1)
        bne   $t0, $s5, Exit
        addi   $s3, $s3, 1
        j     Loop
Exit:  ...
```



Basic Blocks

- A basic block is a sequence of instructions with
 - No embedded branches (except at end)
 - No branch targets (except at beginning)



- A compiler identifies basic blocks for optimization
- An advanced processor can accelerate execution of basic blocks



More Conditional Operations

- Set result to 1 if a condition is true
 - Otherwise, set to 0
- `sl t rd, rs, rt`
 - if ($rs < rt$) $rd = 1$; else $rd = 0$;
- `sl ti rt, rs, constant`
 - if ($rs < \text{constant}$) $rt = 1$; else $rt = 0$;
- Use in combination with `beq`, `bne`

```
sl t $t0, $s1, $s2    # if ($s1 < $s2)
bne $t0, $zero, L      # branch to L
```



Branch Instruction Design

- Why not blt, bge, etc?
- Hardware for $<$, \geq , ... slower than $=$, \neq
 - Combining with branch involves more work per instruction, requiring a slower clock
 - All instructions penalized!
- beq and bne are the common case
- This is a good design compromise



Signed vs. Unsigned

- Signed comparison: `sl t, sl ti`
- Unsigned comparison: `sl tu, sl tui`
- Example
 - `$s0 = 1111 1111 1111 1111 1111 1111 1111 1111`
 - `$s1 = 0000 0000 0000 0000 0000 0000 0000 0001`
 - `sl t $t0, $s0, $s1 # signed`
 - $-1 < +1 \Rightarrow \$t0 = 1$
 - `sl tu $t0, $s0, $s1 # unsigned`
 - $+4,294,967,295 > +1 \Rightarrow \$t0 = 0$

