MIPS Decision Instruction

beq register1, register2, Label1

beq is “Branch if (registers are) equal”

- Same meaning as (Using C):
- If (register1 == register2) goto Label1

bne register1, register2, Lable1

bne is “Branch if(registers are) not equal”

- Same meaning as (using C):
- If (register1 != register2) goto Label1

Called conditional branches

MIPS Goto Instruction

Unconditional branch

j label

Called a jump instruction: jump (or branch) directly to the given label without needing to satisfy any condition

Same meaning as (using C): goto label

Stank C

C Decisions: if Statements

2 kinds of if statements in C

If (condition) clause

If (condition) clause1 else clause2

Rearrange 2nd if into the following:

If (condition) goto Lable1;

clause2;

- goto Label2;

Label1: clasue1;

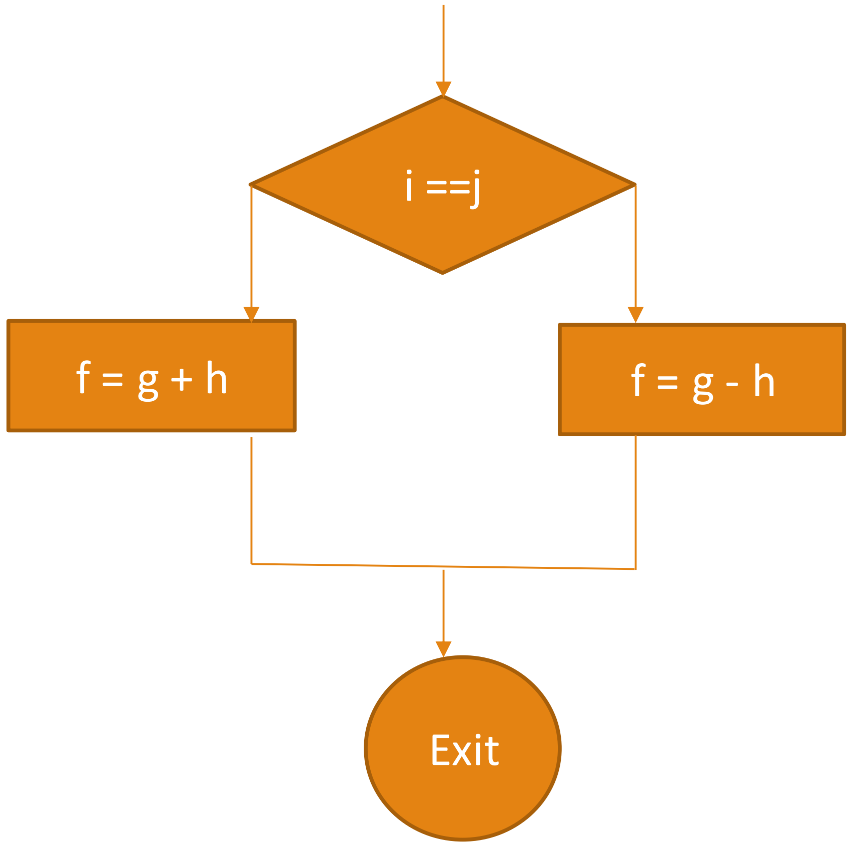
Label2: DoSomething;

Compiling C if into MIPS

Compile by hand

- If (i == j) f = g + h;
- else f = g – h;

- Use this mapping:
- f:	$s0
- g:	$s1
- h:	$s2
- i:	$s3
- J:	$s4

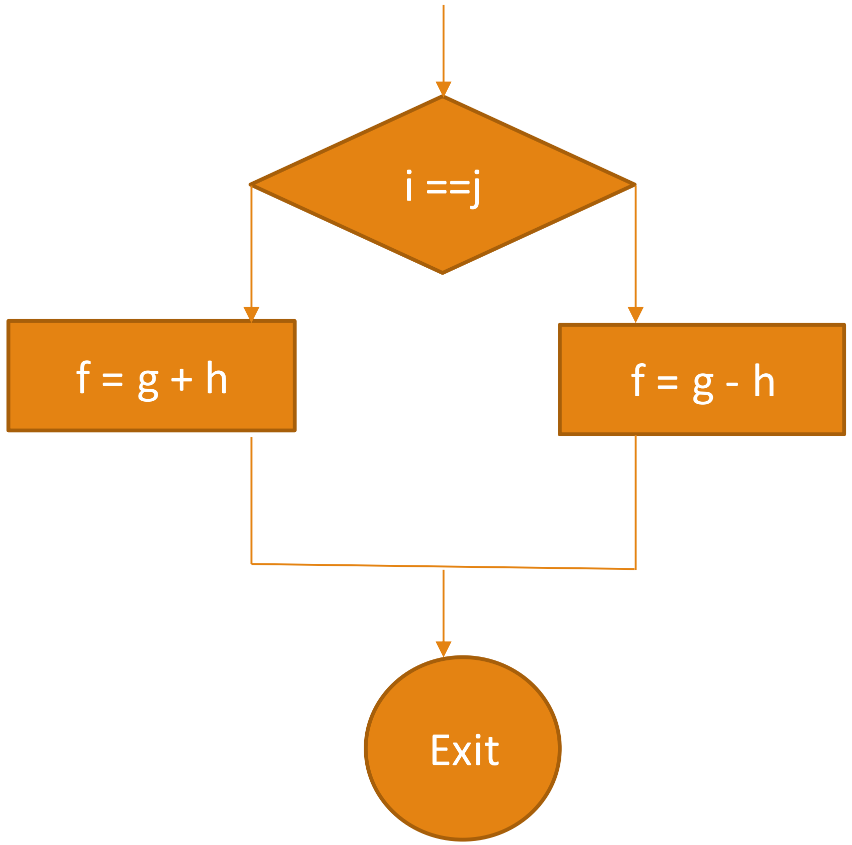


Compiling C if into MIPS

Compile by hand

- If (i == j) f = g + h;
- else f = g – h;

- Final compiled MIPS Code:
- beq $s3, $s4, True		# branch i == j
- sub $s0, $s1, $s2		# f = g – h
- j Fin				# goto Fin
- True: add $s0, $s1, $s2
- Fin:



Peer Instruction

We want to translate *x = *y into MIPS

(x, y ptrs stored in $s0, $s1 repectively)

```
1: add   $so,    $s1,    zero
2: add   $s1,    $s0,    zero
3: lw    $s0,    0($s1)
4: lw    $s1,    0($s0)
5: lw    $t0,    0($s1)
6: sw    $t0,    0($s0)
7: lw    $s0,    0($t0)
8: sw    $s1,    0($t0)
```

a) 1 or 2
b) 3 or 4
c) 5 -> 6
d) 6 -> 5
e) 7 -> 8