COMPUTER ORGANIZATION (IS F242)

LECT 25: MIPS ARCHITECTURE

Jump

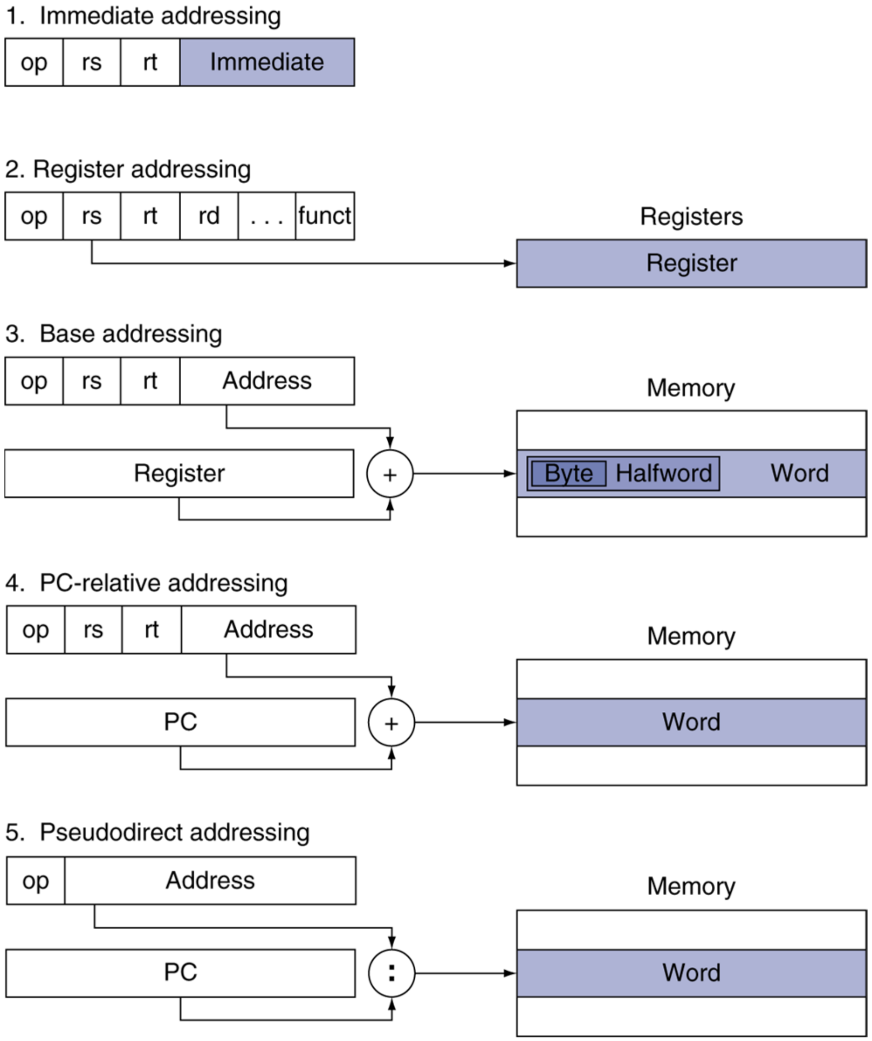
- j imm # Jump absolute
- jal imm # Jump and link ($ra ← PC)
- jr rs # Jump register (PC ← rs)
- jalr rs, rt # Jump register and link (rt ← PC, PC ← rs)
- All jumps are absolute
  - 26 bits absolute address
  - 32 bits??
- All branches are relative to PC
  - 16 bit signed offset

Compute the target Address

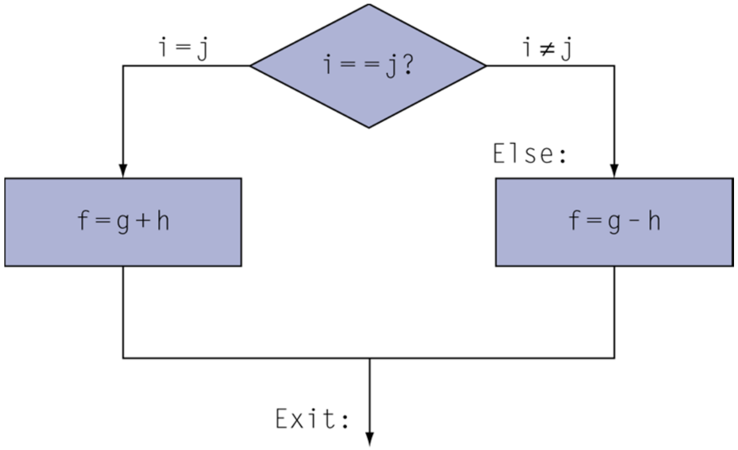
- PC relative
  - 16 bit signed offset
  - + or – 2^{15} bytes from the current instruction??
  - newPC = (PC +4) + (Simm << 2)
- Absolute
  - 26 bit address
  - newPC = (PC & 0xF000 0000) | (Uimm << 2)
  - Address boundary: 256 MB

- j target (2, target)
  - Unconditional Jump
- jal target (3, target)
  - Jump and link
- jalr rs, rd (0, rs, 0, rd, 0, 9)
  - Jump and link register
- jr rs (0, rs, 0, 8)
  - Jump register

Addressing Mode Summary



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

Procedure Calling

- Steps required

1. Place parameters in registers
2. Transfer control to procedure
3. Acquire storage for procedure
4. Perform procedure’s operations
5. Place result in register for caller
6. Return to place of call

Register Usage

- $a0 – $a3: arguments (reg's 4 – 7)
- $v0, $v1: result values (reg's 2 and 3)
- $t0 – $t9: temporaries
  - Can be overwritten by callee
- $s0 – $s7: saved
  - Must be saved/restored by callee
- $gp: global pointer for static data (reg 28)
- $sp: stack pointer (reg 29)
- $fp: frame pointer (reg 30)
- $ra: return address (reg 31)

Leaf Procedure Example

- C code:

```
int leaf_example (int g, int h,
                       int i, int j)
{ int f;
  f = (g + h) - (i + j);
  return f;
}
```

  - Arguments g, …, j in $a0, …, $a3
  - f in $s0 (hence, need to save $s0 on stack)
  - Result in $v0

Leaf Procedure Example

MIPS code:

```
leaf_example:
addi $sp, $sp, -4
sw   $s0, 0($sp)          Save $s0 on stack

add  $t0, $a0, $a1
add  $t1, $a2, $a3        Procedure body
sub  $s0, $t0, $t1

add  $v0, $s0, $zero      Result

lw   $s0, 0($sp)
                          Restore $s0
addi $sp, $sp, 4

jr   $ra                  Return
```