CSSE232
Computer Architecture

Logic and Decision Operations

Class status

- Reading for today:
  - Sections 2.6-2.7
- Due today
  - HW0
- Lab 0 status?

Outline

- Logical operations
- Shift operators
- Pseudo instructions
- Immediates
- Alignment
- Conditionals
- Loops

Logical Operations

Operation	C	Java	MIPS
Shift left	<<	<<	`sll`
Shift right	>>	>>>	`srl, sra`
Bitwise AND	&	&	`and, andi`
Bitwise OR	|	|	`or,` **`ori`**
Bitwise NOT	~	~	`nor`

- Useful for extracting and inserting groups of bits in a word

Shift Operations

- shamt: how many positions to shift
- Shift left logical
  - Shift left and fill with 0 bits
  - sll by *i* bits multiplies by 2^i
- Shift right logical
  - Shift right and fill with 0 bits
  - srl by *i* bits divides by 2^i (unsigned only)

Shift right arithmetic : shift right, fill with sign bits

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Logical Operations

- Logical immediate instructions
  - ori
  - andi
  - xori

- How big are immediate values?

Immediates

- What do you expect?
  addi $t0, $zero, 1

- What about
  addi $t0, $zero, -1

- What about
  ori $t0, $zero, 0x8080

Immediates

- Zero extend or sign extend?

- Arithmetic instructions are sign extended
- Logical instructions are zero extended

Pseudo Instructions

- The assembler is a program that translates
  - add $t0, $t0, $t1
- Into this
  - 0x01094020
- It will also replace 'pseudo instructions' with real instructions
  - lab0, p2.asm: li $t3, 0x12340028

Big Immediates

- What if we need more than 16 bits?

Big Immediates

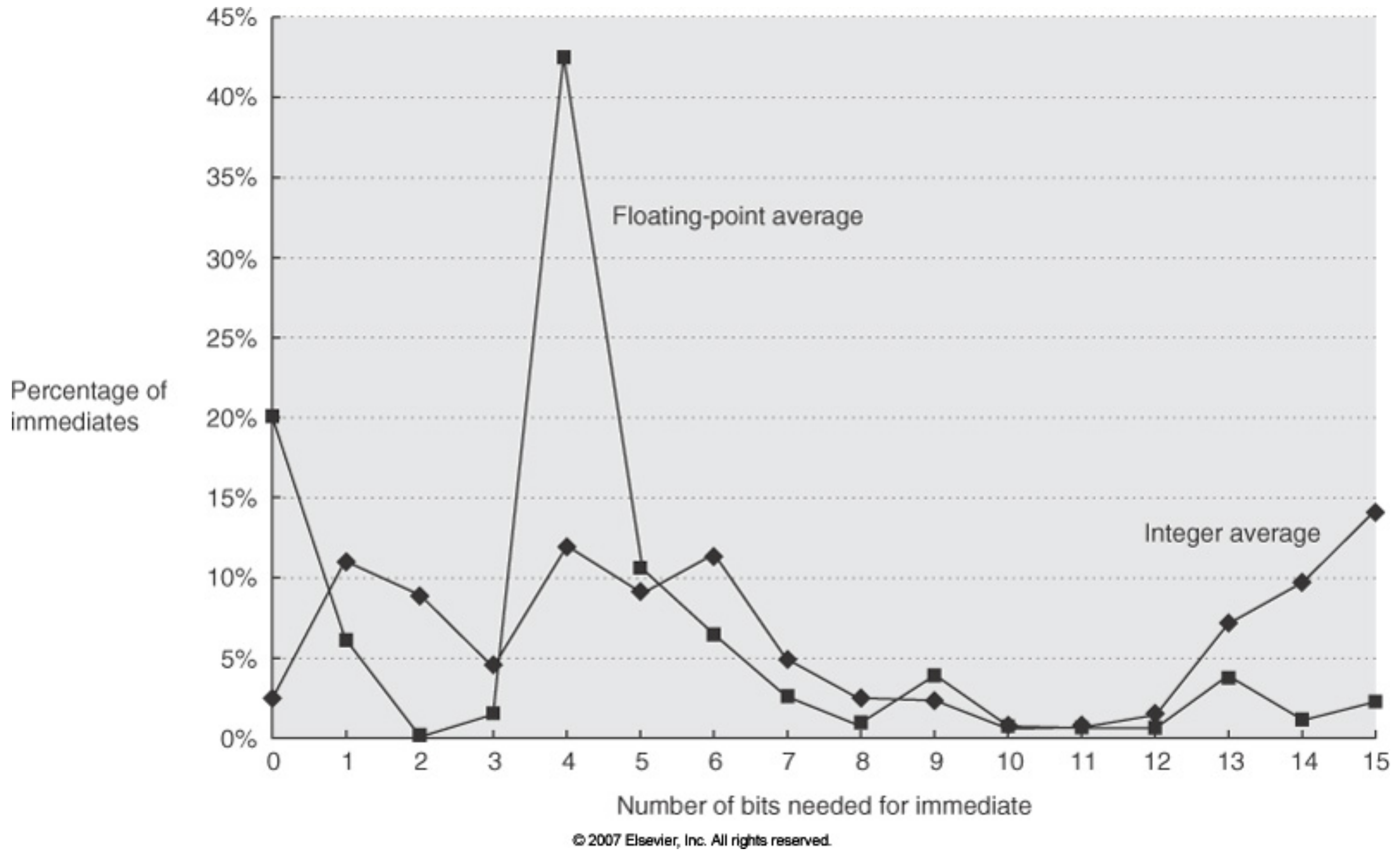
- What if we need more than 16 bits?
  - li : pseudo instruction, composed of…
    - lui
    - ori
- What is the advantage of using a pseudo instruction?
- What is the disadvantage?

Big Immediates

- Is lui a good compromise?

Big Immediates

- Is lui a good compromise?



MIPS alignment restrictions

- The starting address has to be divisible by 4
  - Different from x86 – load from anywhere
- Simple – makes wiring easier

- Big Endian vs Little Endian
  - 4 bytes in a word
  - Which byte is on the big end vs the little end?
  - MIPS – Big Endian
  - X86 – Little Endian
  - Problems when passing data from different machines

Big Endian and Networking

- Convert between host and network format
  - Host format is processor specific
  - Network format is Big Endian
- htons
- htonl
- ntohs
- ntohl

Conditional Operations

- Branch to labeled instruction if 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
- Labels do not have to be capitalized

Basic branching

- C code
  ```
  if(a == 0)
        a = a + 1;
  a = a + 1;
  ```
- MIPS assembler? (a in $t0)

Basic branching

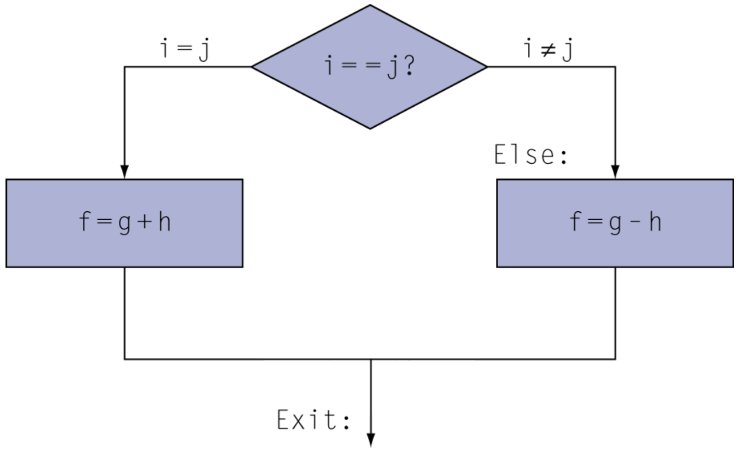
- C code
```
if(a == 0)
      a = a + 1;
a = a + 1;
```
- MIPS assembler? (a in $t0)
```
      bne $t0, $zero, L
      add $t0, $t0, 1
L:    add $t0, $t0, 1
```

More complex if

- 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:  …
```

Conditionals in ARM

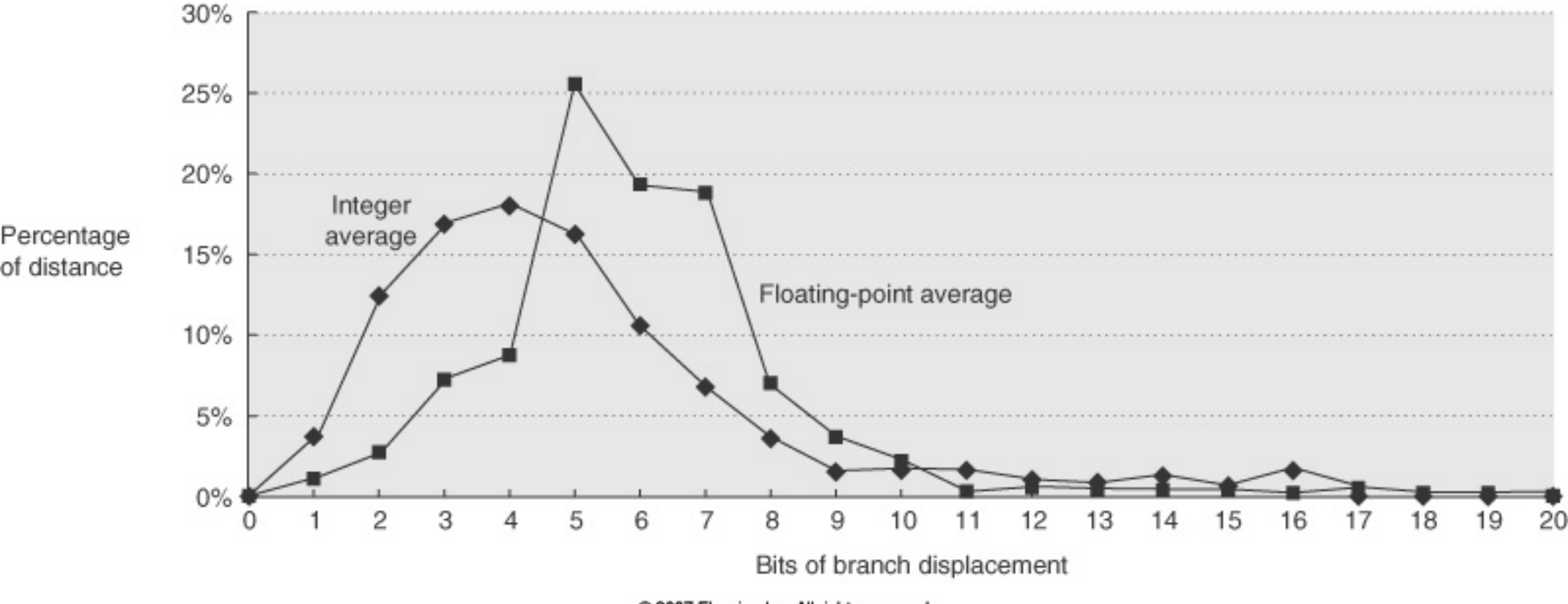
- Every instruction encoding in ARM includes a 4-bit field that represents a condition code.

Conditional jumps

- I-type instructions
- Immediate field holds branch target
  - Is 16 bits enough?

Conditional jumps

- I-type instructions
- Immediate field holds branch target
  - Is 16 bits enough?



- C code:

```
while (n != 0) {
 n--;
}
```

- Compiled MIPS code (if n in $t0):

```
LOOP: beq   $t0, $0, DONE
      addi  $t0, $t0, -1
      j     LOOP
DONE: …
```

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: …
```

More Conditional Operations

- Set result to 1 if a condition is true
  - Otherwise, set to 0
- slt rd, rs, rt
  - if (rs < rt) rd = 1; else rd = 0;
- slti rt, rs, constant
  - if (rs < constant) rt = 1; else rt = 0;
- Use in combination with beq, bne

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

Review and Questions

- Logical operations
- Shift operators
- Immediates
- Alignment
- Conditionals
- Loops