COMPUTER ORGANIZATION AND DESIGN
The Hardware/Software Interface
RISC-V Edition

Chapter 2

Branches and Jumps

These slides are based on the slides by the authors. The slides doesn't include all the material covered in the lecture. The slides will be explained, modified, and sometime corrected in the lecture.

Conditional Operations

§2.7 Instructions for Making Decisions

- Branch to a labeled instruction if a condition is true
  - Otherwise, continue sequentially
- `beq rs1, rs2, L1`
  - if (rs1 == rs2) branch to instruction labeled L1
- `bne rs1, rs2, L1`
  - if (rs1 != rs2) branch to instruction labeled L1

Conditional Operations

Imm[12|10-5]	rs2	rs1	funct3	Imm[4:1|11]	opcode
7 bits	5 bits	5 bits	3 bits	5 bits	7 bits

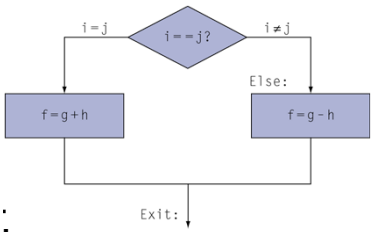
- `beq rs1, rs2, L // if(R[rs1]==R[rs2]) PC=PC+{Imm,1b'0}`

Compiling If Statements

- C code:

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

  - f, g, … in x19, x20, …



- Compiled RISC-V code:

```
      bne x22, x23, Else
      add x19, x20, x21
      beq x0,x0,Exit // unconditional
Else: sub x19, x20, x21
Exit: …
```

Assembler calculates addresses

Compiling Loop Statements

- C code:

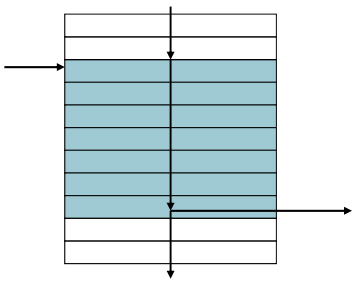
```
while (save[i] == k) i += 1;
```

  - i in x22, k in x24, address of save in x25
- Compiled RISC-V code:

```
Loop: slli x10, x22, 3   //x10=i×8
      add  x10, x10, x25 //x10=save[i]
      ld   x9, 0(x10)    //x9=save[i]
      bne  x9, x24, Exit //save[i]==k?
      addi x22, x22, 1   //i++;
      beq  x0, x0, 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

- `blt rs1, rs2, L1`
  - if (rs1 < rs2) branch to instruction labeled L1
- `bge rs1, rs2, L1`
  - if (rs1 >= rs2) branch to instruction labeled L1
- Example
  - if (a > b) a += 1;
  - a in x22, b in x23

```
bge  x23, x22, Exit      // branch if b >= a
addi x22, x22, 1
Exit:
```


Signed vs. Unsigned

- Signed comparison: blt, bge
- Unsigned comparison: bltu, bgeu
- Example
  - x22 = 1111 1111 1111 1111 1111 1111 1111 1111
  - x23 = 0000 0000 0000 0000 0000 0000 0000 0001
  - `x22 < x23 // signed`
    - –1 < +1
  - `x22 > x23 // unsigned`
    - +4,294,967,295 > +1

Bounds Check

- We can check index out of bounds for array by treating signed numbers as unsigned
  - Check for $0 \leq x < N$
  - bgeu x20, x10, IndexOutOfBounds
  - x20 is x and x10 is N (branch if x20> x11 OR x20 is negative)
- By definition, any –ve number is greater than any +ve number if treated unsigned (MSB=1 for neg, MSB = 0 for +vbe)

Memory Layout

- Reserved (OS)
- Text program
- Static data: global variables, static variables (in C) constant arrays and strings
- Dynamic data: heap (malloc)
- Stack: automatic storage (local to the function)