# ECE222 Chapter 2 Notes

## Instruction Set Architecture

- **Instruction set architecture (ISA):** Defines the interface between hardware and software.

- RISC-V instructions are 32-bits (instruction[31:0]).

- RISC-V assembly uses 64-bit registers (double word) and 32-bit word registers.

- There are 32 registers (x0-x31), with x0 always being zero.

- Arithmetic operations require data to be in registers.

- "Less frequently used" variables are "spilled" into memory.

- Registers are faster and more energy-efficient than memory.

- RISC-V has a 16-bit instruction set (RISC-V compressed) for embedded applications with code size constraints.

- Memory is byte-addressable with little-endian byte ordering.

- RISC-V uses a Harvard architecture with separate instruction and data caches.

## Instruction Formats

- **R-type instructions:** Use three register operands (2 sources and 1 destination).

- **I-type instructions:** Use two register operands and a 12-bit immediate value.

- **S-type instructions:** Use two register operands and a 12-bit immediate value for stores.

- **SB-type instructions:** Conditional branch instructions with PC-relative addressing.

- **U-type instructions:** Upper immediate format for adding an immediate value to PC.

- **UJ-type instructions:** Unconditional jump instructions for jumps and links.

## Examples

- R-type instruction example: `add x5,x6,x2` is coded as 0000000 00010 00110 000 00101 0110011.

- I-type instruction example: `ld x9, 64(x22)` is coded as 0000 0100 0000 10110 011 01001 0000011.

- S-type instruction example: `sd x9, 240(x10)` is coded as 0000111 01001 01010 011 10000 0100011.

- SB-type instruction example: `bne x10, x11, 2000` is coded as 1100000 01011 01010 001 11110 1100011.

- U-type instruction example: `auipc x1, 1000` is coded as 0000000 10001 00000 11011 0010111.

- UJ-type instruction example: `jal x11, 2000` is coded as 1000000 01011 00000 11011 1101111.

# Code Examples

## Compilers and Branches

Compilers frequently create branches and labels where they do not appear in the programming language.

## RISC-V Logical Operations

RISC-V logical operations include shift left/right, right arithmetic, bitwise AND/OR/NOT/XOR.

### 0.0.1 Example 1

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

assume f through j correspond to values in registers x19 through x23

bne x22, x23, Else  // go to Else if i != j
add x19, x20, x21  // f = g + h
beq x0, x0, Exit   // go to Exit

Else: sub x19, x20, x21  // f = g - h

Exit:
```

### Example 2

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

assume value of i is in register x22 and value of k is in register x24.
Assume the base of array save[] is in x25.

loop: slli x10, x22, 3     // x10 = i * 8
      add x10, x10, x25    // x10 now has the address of save[]
      ld x9, 0(x10)        // load save[i] into x9
      bne x9, x24, Exit    // if save[i] != k go to exit
      addi x22, x22, 1     // i++
      beq x0, x0, Loop     // go to loop

Exit:
```

## Procedures

Procedures allow programmers to concentrate on just one portion of the task at a time.

### 0.0.2 Procedure Instructions

By convention in RISC-V:

- 8 parameter registers x10-x17 are used to pass parameters or return values.

- One return address register x1 holds the return address to return to the point of origin.

  In RISC-V, there are special procedure instructions:

- **Jump and link instruction (jal):** For procedures, which branches to an address and saves the address of the following instruction (PC+4) to rd=x1.