CSE3666 Major Topics

Numbers, bits, bytes, and ASCII characters

Numbers in different representations: binary, two's complement, hexadecimal, and decimal numbers.

Conversion between different number systems.

Addition/subtraction of numbers in different number systems.

ASCII characters.

Two's complement numbers

Sign extension. Negate a two's complement number.

Bits

Bitwise logical operations.

RISC-V ISA

RISC-V instructions sets.

Main objectives

Describe how RISC-V supports operations in high-level programming languages: arithmetic, logical, memory access, control flow (if-else and loop). Particularly, describe how RISC-V supports if-else, loops, and functions.

Access data (as immediate, in register, and in memory) properly. Particularly, access data in word arrays and strings.

Describe how RISC-V instructions are encoded (transformed into machine code).

Describe how processor execute instructions in machine code (decoding machine code first).

Write programs with RISC-V instructions and in RISC-V assembly code. Memorize a set of core RISC-V instructions.

Describe limitations of individual RISC-V instructions (e.g., number of registers, immediate, branch offset, and offset in jump) and how RISC-V overcomes limitations with multiple instructions.

Describe the design principles of RISC (compared to CISC) and explain how it affects the design of RISC-V ISA.

Logical and arithmetic operations

Write MIPS instructions to perform common operations.

Accessing Data

Describe how data and arrays (e.g., immediate, integers, addresses, characters, word arrays, byte arrays, and strings) are stored in computers.

Describe the range of immediate in RISC-V instructions.

Describe register file and register numbers/names.

Write RISC-V instructions to access data in registers and in memory. Properly specify addresses. Use proper instructions for data of different types (size and sign).

Explain how endianness affects the byte order when data are stored in memory.

Explain how data size, type (signed/unsigned), and endianness affect the result of load instructions.

Control flow

Program counter.

RISC-V support for if-else and loops.

Branch instructions.

RISC-V support for function calls.

RISC-V calling convention. Passing parameters to / returning values from functions. Caller-saved /callee-saved registers.

Stack. Push/pop. Save/restore registers on stack. Allocate storage space on stack. Explain how stack is used in functions.

Instruction Encoding/decoding

Six instruction formats (R, I, S, SB, U, UJ types).

Fields in different instruction formats.

Describe the placement of immediate bits in machine code, for different types of formats.

Describe how assembler place bits in immediate in machine code and how the processor construct immediate when executing instructions.

Given enough information, encode RISC-V instruction with 32 bits and read encoded instructions.

Assembly code

Explain directives for RISC-V assembly code (e.g., .data, .align, .text)

Describe and explain memory layout of a program.

Translate pseudocode or simple C code to RISC-V assembly code.

Read/debug RISC-V code.