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 processors 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 RISC-V 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 machine code (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 C code to RISC-V assembly code.

Read/debug RISC-V code.

Computer Arithmetic

Digital logic design

Construct truth table from specification. Write logic expressions from truth table. Some truth tables have don't care items, in either input or output.

Describe the general structure of a state machine.

Describe the function of commonly used modules (e.g., decoder, multiplexor, ALU, registers).

Implement digital circuits from logic expressions or diagrams, starting from basic gates or using existing modules.

Given a design of digital circuit, describe its behavior/function.

Analyze the timing of digital circuit.

Implement digital circuit in MyHDL.

Integer multiplication and division

Design multiplier ~~and division~~ module.

Describe the behavior of the circuit.

Explain how multiplication ~~and division~~ are done in hardware.

Code with instructions in RISC-V M-extension.

Floating point numbers

Convert real numbers between decimal and binary representation.

Normalized representation of decimal/binary numbers.

Represent numbers in half-precision, single-precision and double-precision formats.

Read half-precision, single-precision and double-precision numbers and represent the values in normalized representation or in decimal.

Describe RISC-V F/D-extension and write assembly code with instructions in the extensions. Load/store floating-point numbers. Compute with floating-point numbers (in floating-point registers). Convert between word and single/double with RISC-V instructions.

Processor

Computer performance

CPU Execution Time and the classic equation.

CPI. Clock cycle time. Clock rate. Speedup. Amdahl's law.

Compare the processor performance.

Single-cycle processor

Design a single-cycle RISC-V processor. Describe the design of every module in the diagram.

Describe how instructions are executed in the single-cycle design. Describe the hardware components required to support each kind of instructions.

Given an instruction and its address, find the signal values in the single-cycle design.

Improve the design to support more instructions. Describe changes in the design and explain why.

Pipeline

What, why, how questions. What is a processor pipeline, why, and how does it work (how can it be done? Particularly, describe the design of the 5-stage pipeline. Explain the data and control signals in pipeline registers. Describe how the pipelined processor executes instructions.

Describe the hazards in the pipeline.

Describe the method for dealing structural hazards in the 5-stage pipeline.

Describe the method for dealing data hazards in the 5-stage pipeline. Explain how the processor determines forwarding.

Explain the methods for handling control hazards. Always flush. Static prediction.

Explain how the processor detects hazards and stalls the pipeline.

Analyze data/control dependency in code.

Write pipeline diagrams.

Analyze the performance of a pipelined processor of different designs (e.g., with forwarding, without forwarding).

Schedule instructions to reduce hazards.

Cache

Concept of cache. What, why, and how. Temporal locality and spatial locality.

Direct-mapped cache

Fields in an address for cache access. The relation of cache configuration and the number of bits in each field.

Cache lookup. How bits in addresses are used in cache lookup.

Diagram of cache. Hardware cost of cache. Total number of bits in cache. Size of comparators (the number of bits). Number and size of MUXes.

Handling read and write. Write-through. Write-back. Write allocate. No Write allocate. Write buffer.

Given a sequence of memory references(requests) and cache contents, determine the outcome of each reference and update cache.

Cache performance

Memory stall cycles. Instruction cache. Data cache. Average memory access time (AMAT).

Impact of memory system on the performance of processors. Compare performance of processors under different configurations of cache.

Types of cache misses.

Methods for improving cache performance (cache size, block size, set associativity, and multi-level). Impact cache size on block size on AMAT.