Spring 2011 Final Exam Review

fr-1

When / Where

Monday, 9 May 2011 15:00–17:00 (3:00 PM–5:00 PM) CDT (Here).

Conditions

Closed Book, Closed Notes

Bring one 215×280 mm note sheet.

Cannot use communication devices.

Format

Two or three or maybe four medium to long problems.

Short-answer questions.

fr-2

Resources

Check course Web page daily for hints, new resources.

Web page: `http://www.ece.lsu.edu/ee4720/index.html`

RSS feed: `http://www.ece.lsu.edu/ee4720/rss_home.xml`

Solved tests and homework:. `http://www.ece.lsu.edu/ee4720/prev.html`

Statically Scheduled MIPS Study Guide:. . .
. . . `http://www.ece.lsu.edu/ee4720/guides/ssched.pdf`

Dynamically Scheduled MIPS Study Guide:. . .
. . . `http://www.ece.lsu.edu/ee4720/guides/ds.pdf`

Cache Study Guide:. . .
. . . `http://www.ece.lsu.edu/ee4720/guides/cache.pdf`

Study Recommendations

Study homework assigned this semester—**test questions often are based on homework.**

Solve **previous test problems**, start with more recent problems.

Memorizing solutions is not the same as solving.

Following and understanding solutions is not the same as solving.

Use the solutions for brief hints and to check your own solutions.

Emphasis

fr-4

Implementation Diagrams and Pipeline Execution Diagrams

They are a *team*, so study them together.

Instruction Use

Should be able to easily write MIPS programs.

Should be able to use MIPS and SPARC instructions in examples.

Not required to memorize instruction names, except for common MIPS instructions.

Branch Predictors

Understand Operation, Determine Prediction Ratios and Warmup

Cache Diagram, Address Bits, and Program

Determine cache structure from diagram. (Line size, etc.)

Determine hit ratio from program.

Topics

fr-5

Introductory Material

ISA v. Implementation.

Factors influencing ISA and implementation.

Benchmark types.

Compiling and Optimization

fr-6

SPECcpu Benchmark Suite

What SPECcpu is supposed to measure. (Potential of new CPU implementations.)

Importance of having tester compile code.

Peak v. Base tuning: code preparation, use of results.

Why should we trust the SPEC rules?

Why should we trust the results of a test?

Benchmark programs (types, how they were selected).

fr-7

Compilers and Optimization

Steps in building and compiling.

Basic optimization techniques, compiler optimization switches.

Profiling.

Compiler ISA and implementation switches.

How programmer typically uses compiler switches (options).

fr-7

fr-8

Control Transfer Instructions: Types, when to use.

Branch, Jump, Jump & Link, Call, Return

Format of displacements in instruction.

Specification of condition: condition code registers or integer registers.

Instruction Coding.

Fixed-length, variable-length, and bundled instructions.

Splitting of opcode field (as in MIPS type-R instructions).

ISA Classifications: RISC, CISC, VLIW

Dependency Definitions

Hazard Definitions

fr-9

ISA Familiarity

MIPS: Read, Write, Implement

SPARC

Read common instructions.

Use of condition codes.

Instruction coding differences.

Register windows.

fr-9

fr-10

MIPS

Classification: RISC

Goals: ISA should allow simple, high-speed implementation.

Instruction types.

Know how to read and write MIPS programs.

fr-11

Statically Scheduled MIPS Implementations

Unpipelined Implementation

Pipelined Implementations

Basic (no bypassing, 2-cycle branch penalty), bypassed, branch in ID.

For a Given Pipelined Implementation

Show pipeline execution diagrams.

Show register contents at any cycle.

Determine control hardware.

Determine CPI.

fr-12

Interrupts and Exceptions and Traps

Difference between interrupt, exception, trap.

Causes of exceptions, role of handler.

Privileged Mode.

Pipeline activity leading to execution of handler.

Trap Table

Precise exceptions, achieving with floating-point operations.

fr-13

Long Latency (FP) Operations

Types of operations. (Floating point and maybe load.)

Degree of pipelining: Initiation interval.

Detecting functional unit structural hazards.

Detecting WB structural hazards: reservation register.

Detecting and handling RAW hazards: ID-stage v. pre-WB stall.

Handling WAW hazards.

fr-14

Dynamic Scheduling

Register Renaming

How renaming allows out-of-order execution.

Where registers get renamed.

Role of register maps.

fr-15

Reorder Buffer

Normal Use

Issue: placement of instructions in reorder buffer.

Completion: updating reorder buffer entry

Commit (retirement): removal from reorder buffer entry.

Recovery

Goal: Undo execution starting at some instruction.

Reasons: exception, branch misprediction.

Steps when register map not backed up.

Steps when register map backed up.

Superscalar and VLIW

fr-16

n-Way Superscalar

Duplication of Resources.

For n instructions / cylce: Fetch, decode, rename, writeback, commit.

For $< n$ instructions: load/store, floating-point units.

Costs: $\propto n$ functional units; $\propto n^2$ bypass, control.

Performance Limiters

Device Limiters: Lower clock with increasing distances.

Aligned groups impose fetch restrictions that reduce fetch efficiency.

More stalls due to data dependencies.

More squashes due to branches.

fr-17

VLIW

Difference with superscalar: instruction bundling.

Dependence information in bundles.

Deeper Pipelining

fr-18

Deeper (Super) Pipelining

Relationship between stage splitting, clock frequency and performance.

Relationship between stage splitting and cost.

Latch setup time.

More stalls due to data dependencies.

Instruction-Level Parallelism versus Explicit Parallelism

fr-19

Instruction-Level Parallelism and Explicit Parallelism definitions

Exploitation of ILP:

Unpipelined: Not at all.

In order of increasing ILP: pipelined, superscalar, dynamically sched.

Costs of exploiting ILP, or trying to exploit ILP that's not present.

Exploitation of Explicit Parallelism

Human or compiler writes code with multiple threads, tasks, processors, etc.

Each runs on different core of multicore chip or multiple chips.

ILP v. Explicit Parallelism

ILP: No need to re-write code, but costly, uses more energy, lower potential maximum performance.

Explicit: Need to re-write code, not all programs helped.

fr-20

Branch Prediction

Bimodal predictor.

Correlated branch predictors: local, global, gshare.

Branch target prediction.

fr-21

Memory

General

Definitions: bus width (w), address space size (a), character size (c).

Connection of memory devices.

Caches

Cache structure, connection of memory devices.

Line size implications.

Computing hit ratio for given program.

Set Associative Caches

Definitions: line (block), index, tag, alignment, associativity

Connection of tag and data memory.

Special Cases: Direct mapped, Fully associative.