

Midterm Review

- ◆ **Last Time**
 - Multiple Cycle CPU and Control
- ◆ **This Time**
 - Brief review of material covered in the past 5 weeks
- ◆ **Reminders/Announcements**
 - TA Office Hours: W9-11, 3337D APM
 - Review Session: 8 – 10(?)pm, 4301 APM
 - Midterm Exam Thursday
 - » Last Name begins with, take exam at:
 - ◆ A - J HSS 1330 (here)
 - ◆ K - Z York 2622

CS 141 Chien

February 15, 2000

Midterm Structure

- ◆ **80 Minute Exam, closed book, calculators not necessary (not allowed)**
- ◆ **A range of questions which cover very basic understanding to more involved problem solving**
- ◆ **Material: Lectures, textbook, sections, homeworks... everything.**
- ◆ **Should be prepared to “deliver solutions”, not to figure out how things work on the exam --- the time is too tight.**
- ◆ **There are no provisions for makeups so don't miss the exam! :-)**

CS 141 Chien

February 15, 2000

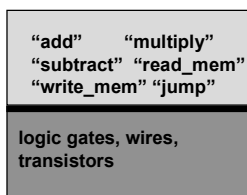
Midterm Coverage

- ◆ Notion of Computer Architecture
- ◆ Performance, Summarizing Performance
- ◆ Assembly Language + Machine Language
- ◆ Basic Processor Implementation

CS 141 Chien

February 15, 2000

Notion of Computer Architecture

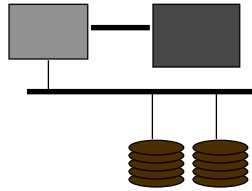


- ◆ Interfaces = abstraction and set of operations
- ◆ Software/Hardware interface, vertical interfaces
- ◆ ISA is critical, as a basis for software portability across machines
- ◆ ISA's form the notion of processor families: 386--486--Pentium, 68000--68020--68040, etc.
- ◆ Computer Architects also put pieces together to make "Computer Systems"

CS 141 Chien

February 15, 2000

Computer SYSTEM Architecture: Horizontal Interfaces



- ◆ **Interfacing parts of the computer system: logical operations (protocols) and electrical signals (signalling)**
- ◆ **Design of interfaces, parts, which shape the SYSTEMs, influencing the performance, cost, and flexibility**

CS 141 Chien

February 15, 2000

Performance

- ◆ **Performance Metrics: purpose, examples**
 - MIPS, MFLOPS, Mhz, Execution time, etc.
 - Pros and Cons of each, how to apply, when to apply
 - Pitfalls and outright cheating
 - Benchmarks and Benchmarking
- ◆ **Comparative Performance Statements**
- ◆ **Amdahl's Law**
- ◆ **Summarizing Performance**
 - Rate and Time measures
 - Arithmetic Mean, Weighted Arithmetic Mean, Harmonic Mean
 - Pitfalls of summarization (conservation of work, normalization), limitations of single number summaries

CS 141 Chien

February 15, 2000

Possible Performance Questions (only suggestive, of course)

- ◆ Describe a scenario, ask you to critique the performance measurement study -- possible pitfalls.
- ◆ Describe pros and cons of several computer performance metrics
- ◆ Given performance measurements (benchmark numbers) give a consistent summary
- ◆ Based on performance measures, summarize performance and compare two systems
- ◆ Identify loopholes in a study due to poor normalization, measurement technique, or describe the right way to do it!

CS 141 Chien

February 15, 2000

Basic Assembly Language

- ◆ **Basic Notions of Assembly Language**
 - Operations
 - Operands
 - Storage classes: Registers, memory , why? (technology and addressing)
 - Load/Store architecture
 - Memory addressing, bytes, chars, words, etc.
- ◆ **More advanced Assembly**
 - Labels and control flow
 - Conditional tests -- performing them and using them to construct conditional execution
 - Branch instructions -- use and limitations
 - Assembly directives (data, instructions, alignment, placement)

CS 141 Chien

February 15, 2000

Assembly (continued)

- ◆ **Addressing types**

- Immediate, register, displacement, PC relative
- Absolute and relative addresses (offsets for various instruction types)
- Field sizes and relation to Machine code Formats

- ◆ **Correspondence of Assembly and Machine code**

- Formats and limitations of the bit field sizes
- Design for fast decoding
- Resolution of instructions into machine code (operand specifiers and labels, mostly)

- ◆ **Long Distance Transfers**

- J and JR ... and...

CS 141 Chien

February 15, 2000

Assembly Language and Procedure Call

- ◆ **Need for Instruction Set support**

- ◆ **JAL, JR basis for procedure linkage and return**

- ◆ **Supporting subroutines (non recursive), and then recursive routines**

- ◆ **Stack discipline, storage for return link, and local state**

- ◆ **Calling conventions and register usage**

- When things go on and come off the stack
- What values will be conveyed in registers where possible

- ◆ **C and Assembly Language correspondence**

CS 141 Chien

February 15, 2000

Full MIPS Calling Conventions

\$0	Constant 0	
\$AT	\$1	Reserved for Asm and OS
\$v0-\$v1	\$2-\$3	Return Values
\$a0-\$a3	\$4-\$7	Arguments
\$t0-\$t9	\$8-\$16, \$24, \$25	Caller saved Registers
\$s0-\$s7	\$16-\$23	Callee saved Registers
\$k0-\$k1	\$26-\$27	Reserved for Asm and OS
\$gp	\$28	Global Pointer
\$sp	\$29	Stack Pointer
\$fp	\$30	Frame Pointer (base of stack frame)
\$31		Return address from jal

CS 141 Chien

February 15, 2000

Enforcing the Register Conventions

- ◆ decide you want to do a procedure call ...
- ◆ Call prologue (save caller saves, args to argregs)
- ◆ JAL (control transfer)
 - Push return link (\$31), allocate local state
 - compute and make recursive calls
 - pop return link, JR \$31
- ◆ Call postlogue (restore caller saves, extract return values)
- ◆ back to whatever you were doing...
- ◆ => conventions ensure interoperability

CS 141 Chien

February 15, 2000

Possible Assembly Language Questions (suggestive, of course)

- ◆ **Basic assembly: understand, write, fix small assembly segments, Control flow, labels, assembly directives: understand, write, fix assembly segments.**
- ◆ **Show their translation to machine code (bits) and understand instruction encoding choices and tradeoffs**
- ◆ **Translate C program to assembly, or write equivalent C program for assembly program**
- ◆ **Procedure call: understand, fix, write an assembly call/return or program (register conventions)**
- ◆ **C and Assembly correspondence on functionality, loops, procedures**

CS 141 Chien

February 15, 2000

Basics of Instruction Execution

- ◆ **Elements of Instruction Execution**
 - Instruction Fetch
 - Instruction Decode, Read Registers
 - Execute operation in the ALU
 - Memory operation, if necessary
 - Write the registers
 - Increment the Program counter
 - Repeat the cycle
- ◆ **Computer: a machine designed to do this repeatedly.**
- ◆ **Differs from other machines because it is programmable -- many instruction seqs possible**

CS 141 Chien

February 15, 2000

Basic Computer Implementation

◆ Single Cycle Implementation

- Instruction Memory, PC, Register File, ALU, Data Memory
- Single set of control settings, data flows through the datapath, settles and end of clock period transitions to the next instruction
- R, I, and J class instructions
- Control logic is pure combinational (implementations)
- Computer = FSM with PC = state
- Advantages: simple control
- Disadvantages: slow (same clock period for all instructions), large hardware requirements (separate memories), low hardware utilization

CS 141 Chien

February 15, 2000

Basic Computer Implementation (cont)

◆ Multiple Cycle Implementation

- Idea: Break execution into clock periods, but how is clock period determined?
- Several sets of control in sequence for the datapath
- Control = Finite State Machine
- Instruction: execution path through a sequence of states
- Advantages: hardware reuse (ALU, Memory), variable instruction execution times (less waste, still some)
- Disadvantages: More complex control, still low hardware utilization (a little better)
- => Generally a necessity for Complex Instruction Sets (CISC's)

CS 141 Chien

February 15, 2000

Possible Questions (suggestive, of course)

- ◆ Identify the control settings and signals to execute a particular instruction
- ◆ Identify the sequence of controls needed to executed instructions in the multicycle datapath
- ◆ Modify the control / datapath to support a new instruction
- ◆ Compare the cost of adding an instruction (complexity and cycle time) to its benefit

CS 141 Chien

February 15, 2000

Summary

- ◆ **Notion of Computer Architecture**
 - technology, influences
- ◆ **Performance**
 - Metrics, Studies, Summaries, Amdahl's Law
- ◆ **Assembly Language Programming**
 - Procedure linkage, stack
 - Register conventions
- ◆ **Basics of Instruction set implementation**
 - Datapath
 - Control
 - Single, Multiple Cycle

CS 141 Chien

February 15, 2000