CSE225 / EEE225 Assembly Language Programming and Microprocessors

Syllabus and Course Information

Dr. Yinong Chen

Catalog Description

Assembly language programming, including input/output programming and exception/interrupt handling. Register-level computer organization, I/O interfaces, assemblers, and linkers. Motorola-based assignments. Lecture, lab. Prerequisites: CSE 100 (or 110 or 200), 120 (or EEE 120).

Prerequisites in topics

The official prerequisite is CSE 100 (or 110 or 200) and CSE120/EEE120 Digital design fundamentals. By topics I expect you have

- Programming skill in a high-level language (C, C++, Java or Pascal), including arrays and indexing, looping, data types, record structures, procedures, conditional and case (switch) branching.
- Binary, octal, hexadecimal, ASCII data representations.
- Two's-complement arithmetic.
- Introductory Boolean logic and minimization techniques.
- Building blocks of digital circuits, AND, OR, NOT, XOR gates, full adder, multiplexor, and decoder.

Textbook (required)

68000 Family Assembly Language, by Alan Clements, Brooks/Cole Publishing, 1994. ISBN: 0534932754

Laboratory Environment

Students will use Metrowerks CodeWarrior Independent Development Environment (IDE), Palm OS and Palm PDA to do assignments. The 68000 assembly programming is embedded in CodeWarrior C environment.

Objectives and Outcomes

1. To develop an understanding of basic computer organization.
   - Students will understand the major components of a computer, the execution steps of instructions and the hardware components used in each step.
   - Students will be able to write pseudo assembly code on different architectures, e.g., accumulator, stack and load-store architectures.
   - Students will understand data representation, instruction set, addressing modes and register organization.
2. To gain an understanding of the relationship between computer hardware and machine code/assembly code.
   - Students will be able to use complex software development tools to assemble programs, test and debug the programs by using breakpoints, single-stepping, and register & memory watches, on a hardware platform or on a simulator.

- Students will be able to apply assembly directives to allocate memory for global variables, and to set the initial addresses for program and data.
- Students can use assembly language to implement flow control constructs (sequential, conditional and iterative).
- Students will understand how the processor identifies different sources of interrupts and exceptions, and invokes the corresponding handler to deal with the interrupt and exception.
- Students will be able to write assembly language programs to read and write the registers in an I/O adapter that controls the communication with I/O devices.

3. To develop skills in modular design and the implementation of software at the assembly level
   - Students will be able to apply subroutines to improve program's modularity, readability and reliability.
   - Students will be able to use the stack to save register contents, to pass parameters to subroutines and to create stack frames for local variables.
   - Students will be able to incorporate an assembly language program as a module of a larger software system written in a high level programming language.
   - Students will be able to design test cases and apply them to fully test the functionality and correctness of their programs.

Topic Coverage

1. Computer organization and architecture: 3 weeks (three lecture hours per week)
   - Data representations, integer and floating-point numbers, arithmetic, arithmetic and logic unit
   - Memory locations, addressing, data units and their sizes: byte, word and long word
   - 68000 organization, processor, memory and I/O, data register and address registers, PC, SP, CCR
   - Introduction to assembly language programming: instructions and assembly directives.
   - Static memory allocation using assembly directives.

2. Addressing modes and program examples using different addressing modes: 2 weeks
   - Register direct, absolute, immediate
   - Register indirect, pre-decrement, post-increment
   - Displacement, indexed, and PC relative addressing
   - Combinations of addressing modes

3. Instruction set and program examples using different instructions in different addressing modes: 2 weeks
   - Instruction formats of some instructions.
   - Execution steps of instructions: fetch, address increment, decode, addressing, fetch operand from register and from memory, perform operation, write result into register and into memory.
   - Data movement, arithmetic / logical rotates, shifts, unconditional branches, absolute and PC relative
   - Testing, conditional branches, absolute and PC relative, conditional and unconditional branches with decrement, indexed jumps, case branching.

4. System Development Process, 1.5 weeks
   - Specification
   - Design Methodology, top-down, modular
   - Implementation
   - Programming testing techniques

5. Subroutines, parameter passing and stack, 1.5 weeks
   - Subroutine instructions: call and return; Different kinds of subroutine calls: consecutive, nested, re-entrant and recursive; Parameter passing methods: call-by-value and call-by-reference
   - Parameter passing facility: Use data register (call-by-value) and address registers (call-by-reference); Use stack for call-by-value and call-by-reference.
   - Local variables on stack: 68000 LINK and UNLK instructions, stack frame for local variables. Nested stack frames for local variable allocation supporting re-entrant and recursive subroutines.
   - Understanding memory allocation in high-level language programming languages: static versus stack memory allocation.

6. Peripheral Programming, 2 weeks
   - Introduction: interfacing processor to devices: synchronous and asynchronous communication, polling versus interrupt driven I/O, and DMA and I/O processors
   - 68000 memory mapped and I/O and interrupt mechanism, and programming 68000 to communicate with devices
   - Serial Interface: The 6850 Asynchronous Communication Interface Adaptor (ACIA)
   - Parallel Interface: The 68230 Parallel Interface/Timer

7. Interrupts and exceptions, 2 weeks
   - Overview of 68000 exception handling facilities
   - Interrupt and exception, privileged status
   - Interrupt and exception handling
   - Palm OS Traps

Cooperation

You are encouraged to cooperate in study group on **preparing** assignments, projects, quizzes and exams. However, anything you turn in must be your own work: You must write up your own solution with your own understanding. If you use an idea that is found in a book or other sources, or that was developed by someone else or jointly with some group, make sure you acknowledge the source and/or the names of the persons in the write-up for each problem.

The instructor and the TAs will CAREFULLY check any possible proliferation or plagiarism. We may also use the software tools like MOSS (Measure Of Software Similarity) to check any assignment that you submitted for grading. The Department of Computer Science and Engineering expects all students to adhere to ASU's policy on Academic Dishonesty. These policies can be found in the Code of Student Conduct:

`http://www.asu.edu/studentlife/judicial/integrity.html`

ALL cases of cheating or plagiarism will be handed to the Dean's office. Penalties include a failing grade in the class, a note on your official transcript that shows you were punished for cheating, suspension, expulsion and revocation of already awarded degrees.