COE 205

Computer Organization and Assembly Language

Computer Engineering Department

College of Computer Sciences & Engineering

King Fahd University of Petroleum & Minerals

Weekly Lecture and Lab Breakdown Proposal for Fall 2006 (Term 061)

Proposed By

Dr. Muhamed F. Mudawar

Monday, March 13, 2006

Week	Lecture	Lab
1	**Introduction to IA-32 architecture** • Basic computer system organization • CPU, memory, and I/O devices • Address bus, data bus, and control bus • Instruction execution cycle • Reading and writing memory • IA-32 general-purpose registers and flags • Instruction pointer and segment registers • Intel processors (from 8086 to Pentium 4) • Modes of operation **Introduction to Assembly Language** • High-level languages • Assembly language • Machine language • Why learn assembly language • Program translation • Tools: editor, assembler, linker, debugger	**Assembly language tools** • Installing MASM 6.15 • Installing 32-bit windows debugger • Installing and customizing the ConTEXT editor • Introductory example: displaying a welcome statement • Using the ConTEXT editor • Assembling, linking, and running a program from the console
2	**Syntax of Assembly Language Programs** • Program template • Directives • Data segment, code segment, and stack • FLAT memory model • Instructions, mnemonics, and operands • Comments • Introductory example • Assembling, linking, and running programs **Data Representation** • Binary, octal, decimal, and hex numbers • Conversion between bases • Signed integers and 2’s complement • Converting signed decimal to/from binary • Storage sizes and ranges • Characters and ASCII table	**Introduction to Assembly Language Programming** • FLAT memory model (32-bit) program template • Writing a 32-bit program that does addition and subtraction using the FLAT memory model • Using the ConTEXT editor to assemble and link the 32-bit program • Using ML and LINK32 commands • Using the MAKE32 batch file to assemble and link 32-bit programs • Using the Windows debugger to view the content of registers

3	**Defining Data and Symbolic Constants** • MASM data types • Data definition statement and initializers • Defining integer data of various sizes • Defining real data of various sizes • Little Endian byte ordering • DUP operator • Defining arrays • Defining strings, null-terminated strings • Visualizing memory allocation • EQU and = directives • Associating symbolic constants with integer expressions and with arbitrary text • Data related operators and directives **Console Input/Output** • Using a simple link library for I/O • Reading a character from standard input • Writing a character to standard output • Reading a string from standard input • Writing a string to standard output • Reading an integer from standard input • Writing an integer to standard output • Other miscellaneous procedures	**Defining Data,** **Symbolic Constants, and** **Data Related Operators** • Defining integer data • Watching variables using the Windows debugger • Multiple initializers and the DUP operator • Watching memory using the Windows debugger • Data-related operators and directives: OFFSET, TYPE, LENGTHOF, SIZEOF, PTR, ALIGN, and LABEL • Symbolic constants and the EQU and = directives • Viewing symbolic constants in the listing (.lst) file
4	**Basic Instructions and Flags** • Data transfer: MOV, XCHG, and LEA • Zero/Sign extension: MOVZX and MOVSX • Register, immediate, and memory operands • Binary addition and subtraction • Carry and overflow • INC, DEC, ADD, SUB, and NEG • Flags: ZF, SF, CF, OF, AF, and PF • LOOP instruction • Writing a loop • Application: generating Fibonacci sequence • Unconditional JMP instruction • IP-Relative displacement • CMP instruction • Selected conditional jump instructions	**Console Input/Output** • Using an external library of procedures for input and output • Writing characters, strings, and integers to standard output • Reading characters, strings, and integers from standard input • Writing a block of memory and registers to standard output • Setting foreground and background colors of text and locating the cursor • Other miscellaneous procedures

5	**Addressing Modes and Arrays** • Addressing modes • Register and immediate operands • 32-bit memory addressing modes • Direct, register indirect, based, indexed • Based-indexed with scale factor • Array indexing and traversal • Using pointers to traverse arrays • Application: copying a string • Application: Summing an array of integers • Two-dimensional arrays • Address computation • Application: sum of a column in a 2D array	**Basic Instructions and Flags** • Data transfer examples using MOV, MOVZX, MOVSX, and XGHG instructions • Addition and subtraction examples using INC, DEC, ADD, SUB, and NEG examples • Using the Windows debugger to view the CF, OF, SF, ZF, AF, and PF flags • LOOP example: generating the Fibonacci sequence • Using the Windows debugger to trace the execution of the LOOP instruction
6	**Conditional Processing** • CMP instruction and flags • Jumps based on specific flag values • Setting and clearing specific CPU flags • Unsigned versus signed comparisons • Jumps based on unsigned comparisons • Jumps based on signed comparisons • JCXZ and JECXZ instructions • LOOPE and LOOPNE instructions • Application: smallest value in an array • Application: searching an array • Application: validating an input string • Application: validating a signed integer • Translating an **IF** statement • Translating a **WHILE** loop • Indirect jump and jump table	**Addressing Modes and Arrays** • Example on addressing modes • Using the Windows debugger to trace memory addressing • Copying a string and tracing its execution • Summing an array of integers and tracing its execution • Using pointers rather than a scaled index to sum an array of integers • Summing a column in a 2D array and tracing its execution

7	**Procedures and the Runtime Stack** • Runtime stack and its applications • PUSH and POP instructions • PUSHFD, POPFD, PUSHF, and POPF • PUSHAD, POPAD, PUSHA, and POPA • Application: reversing a string on the stack • Defining a procedure: PROC and ENDP • Procedure call and return, return address • CALL and RET instructions • Nested procedure calls • Local labels and global labels • Passing arguments in registers • Saving and restoring registers • USES operator • Application: sorting an integer array	**Conditional Processing** • Demonstrating and tracing the execution of the CMP instruction and affected flags • Using conditional jump instructions to find the maximum of three integers • Translating IF statements, WHILE loops, and nested control structures • Demonstrating linear search of an integer array • Demonstrating indirect jump, the jump table, and the translation of a switch statement
8	**Logical and Bitwise Operations** • Logical instructions • AND, OR, XOR, NOT, and TEST • Testing bits in a register • Translating Boolean expressions • Shift instructions: SHL, SHR, SAL, SAR • Rotate instructions: ROL, ROR, RCL, RCR • SHLD and SHRD instructions • Application: binary multiplication • Application: displaying binary bits • Application: isolating a bit string • Application: string encryption	**Procedures and Runtime Stack** • Demonstrating and tracing the PUSH, POP, PUSHFD, POPFD, PUSHAD, POPAD instructions • Demonstrating procedure CALL and RET instructions • Using the Windows debugger to trace the return address on the stack • Demonstrating the saving and restoring of registers • Writing a procedure to sort an array of integers and tracing its execution using the Windows debugger
9	**Advanced Arithmetic** • Integer multiplication: MUL and IMUL • Integer division: DIV and IDIV instructions • CBW, CWD, and CDQ instructions • Divide overflow • Application: string to integer conversion • Application: integer to string conversion • ADC, SBB, STC, and CLC instructions • Extended addition and subtraction	**Logical and Bitwise Operations** • Demonstrating and tracing AND, OR, XOR, NOT, and TEST instructions • Demonstrating and tracing SHL, SHR, SAL, SAR, ROL, ROR, RCL, RCR, SHLD, and SHRD instructions • Writing and tracing a procedure to do string encryption • Writing and tracing a procedure to display a 32-bit register in hexadecimal

10	**Advanced procedures** • Stack parameters • ESP and EBP registers • Accessing parameters on the stack • Allocating local variables on the stack • Accessing local variables on the stack • Stack frames • LOCAL directive • PROTO directive • INVOKE directive • Passing parameters by value • Passing parameters by reference • Memory models and language specifiers • Creating multi-module programs	**Advanced Arithmetic** • Demonstrating and tracing the MUL, IMUL, DIV, IDIV, and CDQ instructions • Demonstrating and tracing the ADC, SBB, STC, and CLC instructions • Writing and tracing a procedure to convert and display a 32-bit signed integer as a string of ASCII characters • Writing and tracing a procedure to do extended addition on two arrays of double word integers
11	**Interrupts** • Software interrupts • INT instruction • Interrupt vector table • Interrupt processing • Interrupt service routing or handler • Hardware interrupts • Exceptions **16-bit MS-DOS Programming** • 16-bit real-address mode programming • MS-DOS memory map • MS-DOS function calls with INT 21h **BIOS-Level Programming** • Keyboard input with INT 16h • Video programming with INT 10h • Mouse programming with INT 33h **Interrupt Handling** • Writing a custom interrupt handler	**Advanced Procedures** • Demonstrating the LOCAL, PROTO, and INVOKE directives • Tracing parameters and local variables on the stack • Demonstrating passing parameters by value and by reference • Writing a multi-module program

12	**String Processing (optional)** • String Instructions • MOVSB, MOVSW, and MOVSD • CMPSB, CMPSW, and CMPSD • SCASB, SCASW, and SCASD • STOSB, STOSW, and STOSD • LODSB, LODSW, and LODSD • REP, REPZ, and REPNZ prefixes • Direction flag • CLD and STD instructions • Application: copying a string • Application: comparing two strings • Application: scanning for a matching char • Application: trimming a string **IA-32 Memory Management (optional)** • Segmentation and segment registers • Real mode memory architecture • Protected mode memory architecture • Logical addresses and linear addresses • Global Descriptor Table (GDT) • Local Descriptor Table (LDT) • Segment descriptor • Paging, page directory, and page table • Linear to physical address translation • Virtual 8086 Mode	•
13	•	•
14	•	•
15	•	•