

Computer Science and Information Systems
TEXAS A&M UNIVERSITY-Commerce

Study Guide for Computer Organization / Assembly Language
CSci516 Qualifying Examination
Fall 2005

The Computer Organization / Assembly Language (CSCI-516) course was designed to give non-computer science students a minimal background in how a computer is organized and how it functions. Before pursuing a MS in Computer Science, all students without a BS in Computer Science from Texas A & M University Commerce are required to complete one of the following:

1. Complete the CSCI-516 course with a grade of B or better.
2. Complete the Computer Organization / Assembly Language Qualifying examination with a passing grade, which is 80% - B.

If you do not have the required background, save yourself and us time and work; just take CSCI-516 - You will not be able to pass this examination. If you do have a Computer Science background, this document is intended to aid you in preparing for the examination.

This examination is based primarily on Intel 8086 microprocessor. It will test your knowledge of both the chip's basic design and your proficiency at using the 8086 assembly language. The examination will also cover a number other basic computer concepts. The following topics may be covered on the examination:

1. Conversion of numbers (both signed and unsigned) from one base to another. The bases covered could be decimal, hexadecimal, binary, and octal. Also know how to convert a decimal number into Intel's short floating point format.
2. Performing logical operations on binary numbers. The logical operations include AND, OR, XOR, NOT, NAND, NOR, and NXOR.
3. You should know the stack operation. This includes its use with both near and far calls, and the use of the PUSH and POP instructions. Also know how parameters are passed through the stack, both by value and by reference. Know how to use a "Stack Frame" to retrieve the parameters. Also know how to clear the parameters from the stack when you return. Definition of variables.
4. 8086 CPU functions for service of an interrupt.
5. Given a high level language segment (in C) and / or a pseudo-code definition, you will be asked to write a assembly language procedure or short assembly language segment to perform the same function.
6. Know how to start and end a procedure. Also know the rules for saving and restoring registers when you enter and leave a procedure.

7. You should know how the linker assign values to the STACK SEGMENT register and what are the values in the other registers.
8. Know how Micro Operations and Control Signals are used to execute a given instruction. Know the difference between fetch cycle and execute cycles. Given the Micro operations and Control Signals used to execute an instruction, be able to explain how the instruction is executed.
9. Know how do logic diagrams work and perform logic functions. You should also know how to complete these diagrams.
10. You should know the basic assembly instructions and how to use them. Also, it is important to know how to specify values in assembly language in binary, decimal and hexadecimal.
11. Know which registers are 8 bit and which are 16 bit. What are the segment registers and how are they used? What are the index registers and how are they used? What are the special uses of the BX and BP registers?

Floating Point Conversion

The following is for converting Decimal numbers to Intel (IEEE) floating point numbers. Floating point numbers are in a form similar to scientific notation.

The general format of an Inter short format (single precision) floating point number is:

	0	1		8	9				31
S	Exponent				Mantissa				
1	8 bits				23 bits				

S - The sign of the number. 0 = Positive. 1 = Negative.

Exponent - a binary **biased** power of two.

Mantissa - a 24 bit (first bit always 1 and not stored) normalized binary number.

To convert from decimal to floating point, do the following:

1. Insert the sign of the number in the first bit. 0 for positive - 1 for negative.
2. Convert the absolute value of the decimal number to binary.

Example: 6.625 = 0110.10100000 X 2⁰⁰⁰⁰

3. Normalize the number. This means shifting the binary point until there is one digit to the left of the point.

$$0110.10100000 \times 2^{0000} = 1.10101000 \times 2^{00000010} \quad (2^2)$$

4. Bias the exponent. An 8 bit exponent can represent a value from 0 to 255. We wish the exponent to be in the range of -128 to +127; therefore we will let a value of 127 represent an exponent of 0. The value we get is known as a biased exponent. We obtain this value by adding 127 to the value of the exponent.

$$\begin{array}{r} 00000010 \text{ - value 2} \\ + 01111111 \text{ - value 127} \\ \hline 10000001 \text{ - the biased exponent} \end{array}$$

5. We are now ready to assemble our floating point number.

0	10000001	101010000000000000000000
---	----------	--------------------------

Insert the sign from step 1. - 0 for a positive number.

Insert the biased exponent from step 4. - 10000001

Insert the mantissa from step 3. Take the normalized number and store the digits to the **right** of the binary point. The 1 to the left of the binary point is always there and is not actually stored as part of the number. Add trailing 0s as necessary.

If you wish to display this as a hex number - it would look like this:

Binary - 0100 0000 1101 0100 0000 0000 0000 0000
 Hex - 4 0 D 4 0 0 0 0

Other examples:

-0.75 sign = 1 number = 0.110000
 normalized = 1.1000000×2^{-01} or 1.1000000×2^{FF}
 Biases the exponent = 1111 1111
 +0111 1111
 (ignore carry out) 0111 1110

1	0111	1110	100	0000	0000	0000	0000	0000
---	------	------	-----	------	------	------	------	------

Hex - B F 4 0 0 0 0 0

0.75 sign = 0 number = 0.110000
 normalized = 1.1000000×2^{-01} or 1.1000000×2^{FF}
 Biases the exponent = 1111 1111
 +0111 1111
 (ignore carry out) 0111 1110

0	0111	1110	100	0000	0000	0000	0000	0000
---	------	------	-----	------	------	------	------	------

Hex - 3 F 4 0 0 0 0 0

0.0625 sign = 0 number = 0.0001000
 normalized = 1.0000000×2^{-04} or 1.0000000×2^{FC}
 Biases the exponent = 1111 1100
 +0111 1111
 (ignore carry out) 0111 1011

0	0111	1011	000	0000	0000	0000	0000	0000
---	------	------	-----	------	------	------	------	------

Hex - 3 D 8 0 0 0 0 0

9.125 sign = 0 number = 1001.0010
 normalized = 1.0010010×2^{03}
 Biases the exponent = 0000 0011
 +0111 1111
 (ignore carry out) 1000 0010

0	1000	0010	001	0100	0000	0000	0000	0000
---	------	------	-----	------	------	------	------	------

Hex - 4 1 1 2 0 0 0 0

Floating Point Practice Problems

Convert the Following Decimal numbers to Floating Point.

- | | |
|------------|------------|
| 1) 2.5 | 2) -2.0 |
| 3) 20.125 | 4) 265.625 |
| 5) -0.375 | 6) 0.0625 |
| 7) 0.6875 | 8) -0.8125 |
| 9) -20.125 | 10) 244.75 |

Answers

- | | |
|----------------------|----------------------|
| 1 - 4020 0000 | 2 - C000 0000 |
| 3 - 41A1 0000 | 4 - 4384 D000 |
| 5 - BEC0 0000 | 6 - 3D80 0000 |
| 7 - 3F30 0000 | 8 - BF50 0000 |
| 9 - C1A1 0000 | 10- 4374 C000 |