CSCI 360 **Midterm Exam Study Guide** **Summer 2020**

The Midterm Exam will be given on Blackboard and is worth 100 exam/quiz points. It covers material from Sections 1 (Binary and Hexadecimal Numbers) through 15 (Immediate Byte Instructions) and Assignments 1 (First Mainframe Program) through 4 (ABENDs and Dump Reading).

Review questions are available in Study Guides & Quizzes on Blackboard. Please take some time to look at them, as they are representative of what will be on the exam.

I. Binary and Hexadecimal Representation and Arithmetic

Your understanding of the binary and hexadecimal systems is crucial when programming in Assembly language.

It is advisable to know how to use the hexadecimal and decimal conversion chart on pages 39 and 40 of the Reference Summary.

- Know direct hexadecimal to binary conversion (`X'A' = B'1010'`)
- Be able to compute the two's complement of binary and hexadecimal integers.
- Be able to perform addition and subtraction of binary and hexadecimal numbers.
- Be able to convert hexadecimal and binary numbers to decimal, and vice-versa.
- Be familiar with the concept of overflow. Know how to detect conditions that result in overflow.

II. Memory Organization

It is important to know how main memory is organized. In some cases you must organize and access your memory according to specific rules, otherwise your programs will not function correctly.

- Know the basic units of memory and their addressibility: 1 Double word = 2 fullwords = 8 bytes = 64 bits.
- Know when data must align to a fullword boundary.
- Know how to use and the difference of the `DC` and `DS` statements. Know which data units will force alignment to which boundaries.
- Know what will cause, and how to debug a specification exception (`0006`).

III. Dumps

When programming and debugging on the machine level, the truth is always in the dump. Do not rely on what the Assembler was 'supposed to do'. The only way to get to the root of a problem is to look in the dump. It is important then that you are familiar with the format of the dump and how to read it.

- Know the basic format. How to find the contents of memory locations and the registers.
- By using the listing, be able to locate instructions and data encoded by the Assembler.
- Be able to interpret the Program Status Word (PSW).
- Know how to compute the instruction length code, condition code, and ABENDing instruction length.
- Given only the PSW, know how to find an ABENDing instruction.

IV. Addressing Storage

One of the features provided by the Assembler is the use of labels, which greatly simplify the task of referencing memory throughout our program. However, it is important to know how the addresses are stored and determined in our instructions. There are no labels in the dumps, so we must be able to find an address given its base-index-displacement (`D(X,B)`) or base-displacement (`D(B)`)values.

- Given a base register, an index register, and a displacement and their values, be able to compute the absolute address.
- Given a base register and a displacement and their values, be able to compute the absolute address.
- Know how the Assembler uses the USING statement to determine the register to use as a base register.
- Know what causes, and how to debug protection exceptions (`0004`), and addressing exceptions (`0005`).

V. Instruction Encoding and Decoding

The basic job of the Assembler is to take your instructions and encode them into their machine-language form. You should be familair with how the Assembler does this encoding, and especially how to do the reverse, known as decoding. That is, given the machine-language code, how to convert back to the explicit Assembler. This is important when debugging from a dump.

- Know the basic `RR`, `RX`, `SS` (for `MVC` and `CLC`) and `SI` (for `MVI` and `CLI`) instruction formats.
- Know how to encode and decode the instructions we have covered.
- Know what causes, and how to debug, an operation exception (`0001`).

VI. Important Instructions

Of course, all Assembler language programmers must be familiar with the instruction set of the computer they are programming.

- Be familiar with the operation of the instructions we have covered up through week 3 of the summer session:
  `A`, `AR`, `S`, `SR`, `L`, `LR`, `ST`, `C`, `CR`, `BC`, `BCR`, `M`, `MR`, `D`, `DR`, `LA`, `LTR`, `LPR`, `LNR`, `LCR`, `BCT`, `BCTR`, `MVC`, `CLC`, `MVI`, `CLI`.
- Know how the multiply and divide instruction use an even/odd register pair to perform their functions.
- Don't forget the eXtended psuedo-instructions: `XREAD`, `XPRNT`, `XDECO`, `XDECI`.
- Know which instructions will set the condition code. Remember: the X-instructions won't be in your Reference Summary!
- Be able to use the branch on condition instructions, including their extended mnemonics discussed in class.
- Know how to use literals. Don't forget a `LTORG`!
- Know how to specify immediate bytes using `MVI` or `CLI` instructions and how they are encoded.

VII. Basic Program Structure

You are expected to know how to implement, in Assembler, the basic structured programing constructs of higher-level languages. These include:

- Simple conditional statements, i.e., if statements.
- Loops using `BCT` and `BCTR`.

- Know how to read and process an indefinite number of records from a file.
- Translating a simple arithmetic formula into Assembler language statements.

VIII. Miscellaneous

- Know how to use the ***System/370 Reference Summary***, or Yellow Card, as far as what has been discussed in lecture or been required in an assignment.
- Know how to interpret the condition code and made decisions based on the value of the condition code.
- Know what `EQU` does.
- Know how to use extended mnemonics `B`, `BR` and `BZ` and the instructions and mnemonics they replace and are encoded as.

Note: Every attempt has been made to make this study guide comprehensive but note that there may be material you are expected to know that may not be included here.