OBJECTIVES

In this lab you will review the GCPU, a good example of a simple instruction set processor. You will gain practice in both programming a microprocessor and understanding the timing in the execution of an instructions set (through a timing simulation). You will also re-learn the process of converting an assembly language program into machine code.

REQUIRED MATERIALS

- GCPU Documentation
  - *Lab on GCPU from 3701 (Fall 2010)*
  - *Archived G-CPU Quartus files*
  - GCPU documentation distributed in class (also on the web as *Documentation and Design Files*)
- Read/save the following document:
  - *Pre-laboratory Report Guidelines*
  - *Create an ASM CCS Project*
    - *ex0.asm, KG_RAM_Link1.cmd*
  - SPRU430E – the F28335 CPU and instruction set guide

PRELAB REQUIREMENTS

It is required that you make a flowchart or write pseudo-code **before** writing **any** program in this course. This will help you formulate a plan of attack for the code.

PART A. SIMULATING PROGRAM FROM 3701 -

Complete part A **from the 3701 Lab 9 document** (from fall 2010) using the G-CPU assembly language.

PART B. ANOTHER GCPU PROGRAM

Write a program, **using the G-CPU assembly language,** to filter data from an array in memory and to copy that data to a specified memory location. Your program can assume that the data is already placed in memory prior to execution, i.e., your program will not have to insert the array into memory.

The array will be in ASCII code and will contain the data in Table 1. An ASCII table can be found at www.asciitable.com. ASCII is a 7-bit coded version of numbers, letters and symbols. The most significant bit (bit 7) is set to zero in our example and the in given website.

Note that the end of the array is indicated by the "null" character. Your program should filter the data in the array shown in Table 1 such that **only** characters with a hex value less than 0x5B are copied. Your program should end after it copies the "null" character. The data in the original array should **not** be corrupted.

Copy the filtered data into consecutive memory locations starting at address ~~0x0A00~~ 0x1A00.

When you write your program, be sure to make your code modular and use relocatable variables, i.e., make it easy to change the location of both your input and output tables and use equ, dc.b, and ds.b appropriately. Use ds.b for the output table. The given input table has only 21 bytes of data; you may assume that any new input table given will have no more than 256 characters of data. Try to make it easy to change the "null" value and the comparison value (0x5B).

Table 1: Memory Array

Address	Data (ASCII)	Data (Hex)
0x0B00	4	0x34
0x0B01	\	0x5C
0x0B02	7	0x37
0x0B03	4	0x34
0x0B04	]	0x5D
0x0B05	4	0x34
0x0B06	(space)	0x20
0x0B07	I	0x49
0x0B08	S	0x53
0x0B09	n	0x6E
0x0B0A	o	0x6F
0x0B0B	(space)	0x20
0x0B0C	t	0x74
0x0B0D	F	0x46
0x0B0E	^	0x94
0x0B0F	U	0x55
0x0B10	N	0x4E
0x0B11	k	0x6B
0x0B12	y	0x79
0x0B13	!	0x21
0x0B14	NUL	0x00

PART C. CREATE AN ASM CCS PROJECT

Go through the *Create an ASM CCS Project* tutorial. Obtain a screen shot on your laptop of the results of step 11, that **also shows your name** in big letters on the same screen. To do a screen shot in Windows, press Ctrl-PrtScrn (i.e., select Ctrl and PrtScrn at the same time). Copy this screen shot into PowerPoint, Word, or similar; save this document and print it.

PRELAB PROCEDURE

1. It is required that you make a flowchart or write pseudo-code **before** writing any program in this course. This will help you formulate a plan of attack for the code.
2. Create your program (in a file on your computer) using the GCPU instruction set.
3. Hand assemble your code, and create a MIF file for Quartus. The MIF file should be created with a text editor, **not** with MS-Word.
4. Run and simulate your code in Quartus. Print out and annotate key parts of your simulation, indicating what is happening in your program. (You do **not** need to print or annotate the entire simulation.) Be sure to label the ASCII characters in your copied array. Make an archive file of the Quartus project.

5. Answer all prelab questions (including the questions from part A in the 3701 GCPU lab document).
6. Bring the following printed documents to turn in to your TA: part B pseudo-code/flowchart, program, and mif file; annotated simulations from part A and from part B, screen shot from part C; and answers to the prelab questions (both below and in part A of the *Lab on GCPU from 3701* document)
7. Email the following to the class gmail account: your Quartus archive file from part B, screen shot from part C.

Note: Prelab requirements **MUST** be accomplished **PRIOR** to coming to your lab.

PRELAB QUESTIONS

1. What are the ASCII values of the data in the copied array?
2. You can find the F28335 instruction set in SPRU430E (the F28335 CPU and instruction set guide), starting on page 159 (document page 6-1). Find three F28335 instructions that you might have used had you written your program in F28335 assembly. Give a short explanation of what each of these instructions do and the GCPU instructions that they replace.

LAB PROCEDURE

Demonstrate that your program works to the TA. The TA will ask you to change the data file and then re-simulate your program. Be prepared to answer questions about your program.