Pre-Lab 5: Program Memory

**Purpose/Scope:** This lab will involve manually encoding assembly instructions to machine instructions, writing the machine code to the processors memory (RAM), and executing the instruction.

**Concepts:** Encoding machine code from assembly instructions. Decode machine code to assembly instruction.

Answers to pre-lab questions are to be completed before coming to your scheduled lab period, for submittal at the beginning of the lab. Use the guidelines for homework. Also keep a record of your answers to use in the lab.

**Arm and Thumb Instruction Sets:**
The Thumb instruction set is a subset of the most commonly used 32-bit ARM instructions. Thumb instructions are each 16 bits long, and have a corresponding 32-bit ARM instruction that has the same effect on the processor model.

Thumb instructions are translated from assembly instructions to machine code instructions in a more straightforward manner than the normal Arm instructions. Therefore, you will practice encoding the machine instructions from assembly instructions using the subset of Arm instructions known as Thumb instructions.

Please note that during the actual Lab05, you will find that the assembler will generate a mixture of Thumb and normal Arm instructions. Be not alarmed. We are concentrating on Thumb instruction encoding because it is easier. However, the concept is similar for the overall Arm instruction set even though we will not cover encoding 32 bit Arm instructions in this pre-lab.

**References:**

**Thumb Instruction Set:**
Pre-Lab 5

Part 1

Encode the following Thumb instructions to machine code using the reference above. To do the encoding use the appropriate format and details of the instruction operation to build the machine code.

Simply put together the bits that make up the 16 bit Thumb instruction and convert it from binary to hex. You are doing manually what the assembler does to encode Thumb instructions from an assembly instruction! Translate the three assembly instructions into machine code, and the last machine code instruction back into an assembly instruction.

1. Format 2: add r3,r5,r6
2. Format 3: add r7,#145
3. Format 9: ldr r2,[r4,#24]

Decode (machine to assembly translation) the following machine code to Thumb assembly instruction:

4. Format 4: 0x41cb
Part 2

The following will introduce a few more \texttt{gdb} commands that you will need for this lab. Use these commands to accomplish the following:

1. Write the appropriate \texttt{gdb} command below to write a short value to memory location \texttt{0x20000000}.
2. Write the appropriate \texttt{gdb} command below to write a word value to memory location \texttt{0x8000000}.
3. Write the command to verify (by examining) a word value at address \texttt{0x8000000}.
4. Write the command to set the PC counter to \texttt{0x8000000}.
5. Write the command to set \texttt{r2} to 0.
6. Write the command to show the values of all the registers.
7. Write the command to show the dis-assembly of a machine instruction in memory.

That’s all. You should have 11 questions and answers.

\textbf{Useful \texttt{gdb} commands:}

1. Show the current values assigned to registers

   \begin{verbatim}
   (gdb) info reg
   ... 
   (gdb) info reg r2 r3 pc
   ... 
   (gdb) print $r2
   ... 
   \end{verbatim}

2. Assign new values to registers

   \begin{verbatim}
   (gdb) set $r4 = 123
   \end{verbatim}

   \begin{verbatim}
   (gdb) set $pc = 0x20000010
   \end{verbatim}

3. Write a byte to memory \texttt{0x2000000}

   \begin{verbatim}
   (gdb) set *(unsigned char *)0x20000000 = 0xab
   \end{verbatim}

Verify
(gdb) x /1xb 0x20000000

4. Write a 16-bit value to memory 0x20000002

(gdb) set *(unsigned short *)0x20000002 = 0xcdef

Verify

(gdb) x /1xh 0x20000002
0x20000002: 0xcdef

(gdb) x /2xb 0x20000002
0x20000002: 0xef 0xcd

5. Write a 32-bit value to memory 0x20000004

(gdb) set *(unsigned int *)0x20000004 = 0x12345678

Verify

(gdb) x /1xw 0x20000004
0x20000004: 0x12345678

(gdb) x /4xb 0x20000004
0x20000004: 0x78 0x56 0x34 0x12

6. Examine machine instruction

(gdb) set *(unsigned short *)0x20000008 = 0x0a23

(gdb) x /i 0x20000008
0x20000008: 1srs r3, r4, #8

(gdb) set $pc = 0x20000008

(gdb) x /i $pc
0x20000008: 1srs r3, r4, #8