

HOMEWORK ASSIGNMENT #2  
Due Wednesday, February 18<sup>th</sup>, 2009

For each programming exercise, use the sample Keil project on the course web page as your starting point. (See the Using Keil uVision3 Projects link on the course web page for more info on using Keil projects.) Rename the *main.s* file to match the filename specified in the problem. Only add your code to the file between the *main\_loop* label and the **B mainloop** instruction. Do not alter the other source files.

**1. (10 points) ARM7TDMI Operating Modes**

What are the operating modes of the ARM7TDMI (list all 7), and for each operating mode give a short paragraph explaining the motivation for having such a mode (Why did ARM include these exception modes?). Also for each mode list any registers that are banked (unique copy for that mode only).

**2. (15 points) Instruction Encoding and Decoding**

a. Determine what the corresponding assembly language instructions are for the binary instructions below, and describe what the instruction will accomplish. Also write the corresponding RTL statement(s) indicating what the instruction does.

```
0xEBFFFFFFE
0xB0820311
0x00000000
```

b. For the following instructions, explain how they can be encoded to ARM7 instructions, how the encoding would be done, what binary value they would be encoded to, and what limitations there are in each case.

```
MVN R7, #-268435441
MOV R9, #-1
LDR R1, =0xFFC03FFF
LDR R2, =0xFF30CFFF
LDR R3, =(MyLabel)
```

**3. (5 points) ARM7TDMI Flags**

List the flag bits that are present in the CPSR, what each represents, and explain when they are updated.

**4. (20 points) Addressing Modes**

Using the uVision3 sample project provided on the web page as a starting point, create a complete ARM program that has the following constants and variables. Do **not** make any changes to *aduc7026.s* or *exceptions.s*. Rename the *main.s* file to *teamX\_main.s*, where *X* is your homework team number. [Be sure to read through the documentation standards before beginning this assignment.]

- a) Declare a constant *ARRAY\_LENGTH* to be equal to 8.

- b) Allocate a half-word variable *word16* in the code area, and initialize it to the **binary** value 1101 0011 1000 1011 (not an equivalent hex number, but rather make the define in a binary representation).
- c) Allocate a byte variable named *byte0* in the code area, and initialize it to 0xC3.
- d) Allocate a 10-element byte array *byte\_array* in the code area, and initialize it to 1, -2, 3, -4, 5, -6, 7, -8, 9, -10.
- e) Allocate space in the data area for an array of *ARRAY\_LENGTH* halfword variables named *halfword\_array*.

In your program, write code to do the following (in sequence). Determine a reasonable (hopefully minimal) set of instructions to accomplish the required items, being sure to use the required addressing modes. You may add constant data to store variable addresses as required. **Do not use any pseudo-instructions. Use explicit PC relative addressing, and PC relative + DCD when necessary**

- f) Using indirect addressing with R6, do **byte** transfers (as many as required) to set the elements of *halfword\_array* to the values 1, 2,4,8,16,32,64,128. Only set the value in R6 once, and then do not alter the value in R6 to complete this item.
- g) Load the value in *word16* to R7, but use two **byte loads** and any other required code to get the full 32-bit value. Assume that word16 represents a **signed** value.
- h) Load the value in *byte0* into R8, assuming it is an unsigned value.
- i) Load the value in *byte0* into R9, assuming it is a signed value.
- j) Write code that finds the minimum value stored in *byte\_array* and puts the sign-extended result in R10. (Assume that the values are stored in **unsigned** binary form, and that you do not know the values in the array!)
- k) Write code that finds the minimum value stored in *byte\_array* and puts the sign-extended result in R10. (Assume that the values are stored in **2's-complement** form, and that you do not know the values in the array!)

Comment *teamX\_main.s* to indicate the sections of code that correspond to each item in the list.

**Important:** Submit ONLY your program source code file *teamX\_main.s* using your homework team's dropbox in Learn@UW. Also, submit a **paper copy** of *teamX\_main.s* with the rest of the assignment.

### 5. (15 points) JTAG Scan Interface

- a) Describe the capabilities that the JTAG scan chain adds to the Texas Instruments 74BCT8245A as compared to Texas Instruments 74BCT245. List the 4 signals that form the test access port (TAP) and describe their functions. Give a brief description of what a TAP controller is, and how it is operated.

Refer to IEEE standard 1149.1 and the datasheets for the devices. Sections 5-7 and figure B.10 in the standard are particularly helpful to look at, as well as the datasheets. The datasheets are available at <http://www.ti.com>. You can get that standard at IEEE Xplore going through the Wendt library site, or directly at <http://ieeexplore.ieee.org/Xplore/dynhome.jsp>. Going through Wendt ensures that you will not have access problems to the IEEE site.)

- b) Compare the SN74BCT8245A and SN54BCT8245A device characteristics. What are the key differences and similarities? What conclusions can come to about the two parts?

### 6. (15 points) Load/Store and Conditional Instructions

Using the uVision3 sample project provided on the web page as a starting point, rename the *main.s* file to *teamX\_hw2p6.s*, where *X* is your homework team number. Declare 3 word arrays named *src*, *dest1*, and *dest2*, each of which contain 4 elements. The *src* array is to be declared in the code area, and should be initialized to the values 0x00010203, 0x04050607, 0x08090A0B, 0x0C0D0E0F. The *dest* array is to be declared in the data area, and should be uninitialized.

- a) Write code that will copy the data from *src* to *dest1*, to meet the following conditions. When the byte value is less than 0x0A perform a straight copy of the byte. If the byte value is greater than or equal to 0x0A, place its complement in the destination instead. Assume that you do not have any prior knowledge of the source data values. You may use the LDR pseudo-instruction to load the array addresses.
- b) Write code that will copy the data from *src* to *dest2* using load/store multiple instructions. This time it is just a straight copy.

Separate the blocks of code, and clearly comment to indicate the code for a given task. **Make use of conditional execution in your code!**

**Important:** Submit ONLY your program source code file *teamX\_hw2p6.s* using your homework team's dropbox in Learn@UW. Also, submit a **paper copy** of *teamX\_hw2p6.s* with the rest of the assignment.

### 7. (10 points) Memory and Addressing Modes

- a) Refer to the code you wrote for problem 6 above. If you initialize the data in the *dest1/dest2* array, what happens? Does this make sense?
- b) B. Again referring to the code you wrote for problem 6 above, try to use the following pseudo-instructions to load the array addresses instead.
- ```
ADR R0, src
ADR R0, dest1
```

Explain what happened, and why it happened. If a pseudo-instruction works, describe in detail how it actually accomplished the load. If a pseudo-instruction will not work, explain in detail why it does not work (e.g. why is it not possible for it to work?).

### 8. (10 points) Exam Question

Design one original quiz question operating at Bloom's Taxonomy level 3 for any material covered in Module 2. This must test one of the module objectives in a specific problem. Explicitly state which particular objective you are attempting to test. Provide a complete, detailed solution to your question.