ARM ISA and Assembly

Tom Kelliher, CS 220

1 Administrivia

Today's Objectives

1. Begin to familiarize yourself with the ARM ISA.
2. Learn the structure of an ARM program and use Keil uVision4 to assemble and run simple ARM programs.
3. Understand and use ARM data-processing instructions.

Next Up

Read 3.6.

1. Understand and use unconditional and conditional branch instructions, and the compare instruction.
2. Manually compile if/else blocks into ARM assembly.
3. Manually compile for and while loops into ARM assembly.

2 Warm-Up

1. In words, the instruction

   ```
   sub r2, r0, r1
   ```

   (a) Subtracts the contents of r1 from r0, storing the result in r2.

   (b) Subtracts the contents of r0 from r1, storing the result in r2.

   (c) Subtracts the contents of r0 from r2, storing the result in r1.

   (d) Subtracts the contents of r2 from r0, storing the result in r1.

   (e) Is it nap time yet?

2. In this ARM instruction sequence:

```
subs r1,r1,#4
add r2,r2,#4
mul r5,r1,r2
beq error
```

the condition codes tested by the `beq` instruction are a result of the execution of which instruction:

(a) `subs`

(b) `add`

(c) `mul`

(d) None of them because `beq` is an unconditional branch.

3. The following ARM program is invalid because

```
   area foo, code, readonly
   subs r1,r1,#4
   add r2,r2,#4
   mul r5,r1,r2
s  b s
   end
```

(a) The assembler directives aren't capitalized.

(b) The code's starting point isn't indicated.

(c) The code's ending point isn't indicated.

(d) All of the lines have to begin in column 1.

4. In an ARM assembly file, only labels should start on line 1.

   True/False.

3 Problems

1. Follow the *Quick Guide to Using the Keil ARM Simulator* section on pg. 1 of the *Graded ARM Assembly Language Examples* document to create and run an ARM assembly program.

2. Modify the previous program so that register `r2` is incremented by `0xa` in an infinite loop. (Insert another add instruction immediately following the add instruction already in the program, and change the target of the unconditional branch instruction.) Run your program in single step mode to confirm correct behavior.

3. Create a second project, copy the `reverseBytesLab.s` program from the course web site to it, add this program to your project, and complete the two exercises described in the program.