CSE 320 **Fall 2018**

Computer Project #7

Assignment Overview

This assignment develops familiarity with the instruction set architecture of the ARM microprocessor, as well as the format of its machine language instructions.

It is worth 40 points (4% of course grade) and must be completed no later than 11:59 PM on Thursday, 10/25.

Assignment Deliverables

The deliverables for this assignment are the following files:

`proj07.makefile` – the makefile which produces **`proj07`**
`proj07.support.c` – the source code for your support module
`proj07.driver.c` – the source code for your driver module

Be sure to use the specified file names and to submit them for grading via the CSE handin system before the project deadline.

Assignment Specifications

An assembler is a program which converts assembly language statements into machine language instructions; a disassembler is a program which converts machine language instructions into assembly language statements.

1. You will develop a support module which could be used by a disassembler for ARM machine language instructions. The interface to the support module is the following C function:

```
void decode( unsigned int, char[] );
```

empty array

The first argument is the bit pattern for an ARM machine language instruction.

The second argument is the address of an array where function "decode" will store a null-terminated character string representing the machine language instruction.

Function "decode" will accept a 32-bit unsigned integer value and attempt to disassemble it. If the bit pattern corresponds to an ARM machine language instruction from the set defined below, function "decode" will produce a human-readable representation of the instruction and store it in the array (the second argument). Otherwise, it will produce an appropriate warning and store it in the array.

The support module will consist of function "decode" and any additional helper functions which you choose to implement. The support module will not perform any input or output operations. For example, the functions in the support module will not call function "scanf" or function "printf".

2. You will develop a driver module to test your implementation of the support module. The driver module will consist of function "main" and any additional helper functions which you choose to implement. All output will be appropriately labeled.

Your driver module may not be written as an interactive program, where the user supplies input in response to prompts. Instead, your test cases will be included in the source code as literal constants.

Assignment Notes

1. Your driver module and your support module must be in separate source code files.

2. Your source code must be translated by "gcc", which is a C compiler and accepts C source statements.

3. You must supply a "makefile" (named "proj07.makefile"), and that makefile must produce an executable program named "proj07".

4. Your support module will recognize the ARM data processing instructions, which have the following format:

Bits 31:26	111000
Bit 25	I bit
Bits 24:21	opcode
Bit 20	S bit
Bits 19:16	Rn 0000 mov not used
Bits 15:12	Rd
Bits 7:0	immediate value (when I=1)
Bits 3:0	Rm (when I=0)

All other bits will be 0.

unused bits all 0

S bit 1

The sixteen operation codes are given in the following table:

Opcode	Mnemonic		Opcode	Mnemonic
0000	and		1000	tst
0001	eor		1001	teq
0010	sub		1010	cmp
0011	rsb		1011	cmn
0100	add		1100	orr
0101	adc		1101	mov
0110	sbc		1110	bic
0111	rsc		1111	mvn

compare Rd 0000

mov

5. Your support module will produce a human-readable representation of a machine language instruction which is in the same format as ARM assembly language instructions. Registers will appear as the character “r”, followed by a decimal number between 0 and 15. Immediate constants will appear as the characters “#0x”, followed by a hexadecimal number. For example:

Instruction	Human-readable form
e086a007	add r10, r6, r7
e096a007	adds r10, r6, r7
e286a007	add r10, r6, #0x7
e1520003	cmp r2, r3
e1a01003	mov r1, r3

6. Note that the functions in your support module cannot perform any input or output operations. All communication between the driver module and the support module will be done via the two arguments to function “decode”.