CSCI 2321 (Principles of Computer Design), Spring 2015

Homework 2

Credit: 50 points.

1 Reading

Be sure you have read all assigned sections of Chapter 2 and Appendix A.

2 Notes

- If the assignment asks you to do one or more problems from the textbook, *be sure* you get them from the edition specified in the syllabus; these sets of problems change from edition to edition.
- If a question requires you to do calculations, your odds of getting partial credit are much better if you show enough work to make it clear how you arrived at your answer.

3 Problems

Do the following problems. You may write out your answers by hand or using a word processor or other program, but please submit hard copy, either in class or in one of my mailboxes (outside my office or in the ASO).

1. (5 points) Do problems 2.9 and 2.10 from the textbook.
2. (5 points) Do problems 2.14 and 2.15 from the textbook. ("Type" here means the format of the instruction — R, I, or J.)
3. (5 points) Do problems 2.19.1, 2.19.2, and 2.19.3 from the textbook. Two of the instructions don't really make sense. Which ones, and why not?
4. (5 points) Do problem 2.27 from the textbook.
5. (5 points) Do problem 2.29 from the textbook.

   (*Correction:* There are a couple of instructions in the MIPS code in the textbook that don't make sense. I'm guessing they're typos and that the authors intended to write the following.)

```
        addi    $t1, $0, 0
LOOP:   lw      $s1, 0($s0)
        add     $s2, $s2, $s1
        addi    $s0, $s0, 4
        addi    $t1, $t1, 1
        slti    $t2, $t1, 100
        bne     $t2, $0, LOOP
```

6. (10 points) Do problem 2.31 from the textbook, omitting the part of the problem that asks you to count instructions. (I.e., just provide the MIPS code.)

7. (10 points) Show what would be produced by linking two object files described by the following tables, producing something analogous to the answer to the worked example on pp. 127ff — i.e., show the result of everything the linker must do). Assume that the following sizes:

   - For procedure `A`, a text size of 0x140 and a data size of 0x40.
   - For procedure `B`, a text size of 0x300 and a data size of 0x50.

   and assume the overall layout shown in the figure on p. 104.

   Table for procedure `A`:

Text Segment	Address	Instruction	
	0	`lui $at, 0`	
	4	`ori $a0, $at, 0`	
	...	...	
	0x84	`jr $ra`	
	...	...	
Data Segment	Address	Label	
	0	(`X`)	
	...	...	
Relocation Info	Address	Instruction Type	Dependency
	0	`lui`	`X`
	4	`ori`	`X`
Symbol	Address	Symbol	
	...	`X`	

   Table for procedure `B`:

Text Segment	Address	Instruction	
	0	`sw $a0, 0($gp)`	
	4	`jmp 0`	
	...	...	
	0x180	`jal 0`	
	...	...	
Data Segment	Address	Label	
	0	(`Y`)	
	...	...	
Relocation Info	Address	Instruction Type	Dependency
	0	`sw`	`Y`
	4	`jmp`	`FOO`
	0x180	`jal`	`A`
Symbol	Address	Symbol	
	...	`Y`	
	0x180	`FOO`	
	...	`A`	

4 Programming Problems

Do the following programming problems. You will end up with at least one code file per problem. Submit your program source (and any other needed files) by sending mail to `bmassing@cs.trinity.edu`, with each file as an attachment. Please use a subject line that mentions the course number and the assignment (e.g., "csci 2321 homework 2"). You can develop your programs on any system that provides the needed functionality, but I will test them using the SPIM simulator on one of the department's Linux machines, so you should probably make sure they work in that environment before turning them in.

1. (5 points) Add code to your solution to problem 2.27 to make it a complete program that prompts for values for `a` and `b` and prints the ending values of the elements of `D`, i.e., have the program do the equivalent of the C code

   ```
   for (k=0; k<(4*b); k++)
       printf("%d\n", D[k]);
   ```

   Programs `echo.s` and `echoint.s` on the sample programs page show how to input and output text and integer values.

2. (Optional: Up to 10 extra-credit points.) First do problem 2.37 from the textbook, but using the condition that the input ASCII string is meant to represent a positive hexadecimal integer (so for example "1234" and "5A" are valid inputs, but "hello" is not, nor is "-1"). Then add code to the resulting function so that the result is a complete MIPS program that prompts for an input string, converts it to an integer using your function, and prints the result (in base 10). Programs `echo.s` and `echoint.s` on the sample programs page show how to input and output text and integer values.