School of Computer Science and Information Technology
Computer Systems Architecture (G51CSA) Autumn 2008
Thorsten Altenkirch

Coursework 3 (cw id 119)

Monday, 13 October 2008
Deadline: 20 October 08, 12:00

Collaborating in small groups of up to three students is permitted, but you must implement your own programs (absolutely *do not* copy and paste from others) and provide your own answers where appropriate.

The exercise has to be submitted using the departmental coursework submission system, see
http://support.cs.nott.ac.uk/coursework/cwstud/.

Create a directory `ex03` and put all the files to be submitted (but nothing else) into this directory before submitting the directory.

Multiple submission before the deadline are allowed, only the last one will be taken into account.

1. Decode the following machine code (i.e. translate into assembly language):
   ```
   3401002A
   02328022
   3C041001
   ```
   Store your results in a file called `ex03-1.asm`.

   Hints: Read *Hennessey and Patterson*, pp102–104: '*Decoding Machine Code*' and Appendix A-50, Figure A.10.2 may be helpful.

   - Convert to binary notation
   - Find the main opcode (bits 31-26) field
   - Determine the instruction type, and remaining fields
   - H&P, Appendix A-50, Figure A.10.2 may be helpful

2. Study the following C program:

   ```
   #include <stdio.h>

   int main() {
     int i,j;
     for(i=1;i<10;i++) {
       for(j=0;j<i;j++) {
         printf("*");
       }
       printf("\n");
     }
   }
   ```

Try to predict what it is doing before running it! If you need more background on C, check out the wikibook: *Programming in C* available at http://en.wikibooks.org/wiki/C_Programming.

Now compile and test your program:

- Store the source code in a file called `stars.c`.
- Compile the program using:

  ```
  gcc stars.c -o stars
  ```

- Test your program using:

  ```
  stars
  ```

3. Replace the for-loops by equivalent while-loops (following the method discussed in the lecture). Store your new program as `starsw.c` and compile and test it. It should produce the same output as the previous program.

4. Replace the for-loops by gotos, introducing labels where appropriate (following the method discussed in the lecture). Store your new program as `starsg.c` and compile and test it. It should produce the same output as the previous program.

5. Translate the previous program into MIPS assembly language. In particular you need to:

   - Declare strings in the data section corresponding to the strings used in the C program.
   - Assign registers to the variables used in the C program.
   - Translate the conditional and unconditional gotos of the C program into equivalent MIPS instructions.
   - Replace the calls to `printf` by calls to the `print_string` system call.
   - Finish your program by `jr $ra` or by calling the `exit` system call.

   Call your program `stars.asm` and test it using the SPIM simulator. It should produce the same output as the previous programs in the SPIM console window.