Indian Institute of Technology, Delhi

FALL, 2012

COMPUTER ARCHITECTURE

Homework 1

(DEADLINE: September 4th, 11:59 PM)

NOTE: All answers need to be brief and to the point.
Please make any assumptions that you deem to be reasonable.
Total Marks: 50

Policies:

1. If you are late by n days and m minutes, then we will assume that you are $n + 1$ days late. There is a 10 mark penalty per day.
2. Do your homework in groups of 2 people.

Submission: All submissions will be done through Moodle. `https://jaijaivanti.cse.iitd.ernet.in`

Software Required:

emuArm Emulator

1. emuArm Emulator
   - Download from `http://www.cse.iitd.ac.in/~srsarangi/software.html`
   - To run it : java -jar emuArm.jar
   - Any queries, suggestions or bug reports: Please post it on the course google group: csl211

ARM Tool Chain

Installation of Tools: Learn to use the gnu ARM cross compiler. There are two ways in which you can obtain it.

1. Use the installed tool chain on palasi.cse.iitd.ernet.in :
   - export PATH=/misc/research/teaching/srsarangi/gnuarm-4.0.2/bin:$PATH
   - You should now be able to run arm-elf-gcc
   - To get an assembly file: arm-elf-gcc -S filename.c
   - Compile an assembly file: arm-elf-gcc filename.s
   - To run a file : arm-elf-run ./a.out

CONTINUED

- You can use the command arm-elf-run to run an ARM binary. arm-elf-gdb can be used to debug an ARM binary. However, this only works on Mac OSX. On Linux you can use the program arm-elf-insight, which is a GUI frontend for the ARM debugger.

2. Install on your own:
   - Install Linux on your laptop
   - You can either do a dual boot installation, or install on top of the vmplayer virtual machine. You can download vmplayer from the website of VMWare.
   - Go to www.gnuarm.com
   - Download gnuarm-3.4.3 for 32 bit Linux, or the latest version for 64 bit Linux
   - Run the command : tar -zxf <name of downloaded file>
   - arm-elf-gcc should be in gnuarm-XX/bin

Qemu

1. Qemu is installed on palasi.cse at /misc/research/teaching/srsarangi/qemu-0.14.1/build/bin/qemu-arm
2. You can download qemu from `http://wiki.qemu.org/Main_Page`. Download the latest version, and compile it from source. It should build without any difficulty on Linux/Unix systems. There might be a couple of small problems on Mac systems. However, with some help from Google it should be possible.

How to use qemu:
Running Qemu

- qemu-arm a.out

Debugging with Qemu

- Compile your .s file : arm-elf-gcc -g < .s file >
- qemu-arm -g <port number> a.out (e.g. qemu-arm -g 1234 a.out)
- arm-elf-gdb a.out
  - On the gdb prompt type : target remote localhost:<port number> (e.g. target remote localhost:1234)
  - Your can now start debugging

1. (25 marks)

You need to solve an instance of the matrix chain multiplication problem. See `http://en.wikipedia.org/wiki/Matrix_chain_multiplication`. You need to implement the dynamic programming solution in assembly, and then minimize the number of static assembly instructions. We are only concerned about assembly instructions, not labels, directives, comments, and blank lines.

Input: : You will be given a file with 50 entries. Each line will contain the dimensions of the matrix (row,col) separated by a space. The output will be a parenthesized list. Example:

```
File
------------
3 5
5 7
7 9
```

Example Output : (M1x(M2xM3)). The first row corresponds to matrix, M1, the second to M2, and so on. We will use the character 'x' for multiplication.

2. (25 marks)

Find a fairly non-trivial algorithm for multiplying large integers.
Examples: Toom-Cook algorithm, Schonhage-Strassen algorithm

Implement this algorithm in assembly, and explain why it is better than conventional solutions. Show a live demo to the TAs, and explain all the assembly optimizations. You can choose to either minimize the number of static assembly instructions, or the execution time, or some other metric that makes sense to you.