Electrical and Computer Engineering Department
University of Maryland
College Park, MD 20742-3285
Glenn L. Martin Institute of Technology ♦ A. James Clark School of Engineering
Dr. Charles B. Silio, Jr.
Telephone 301-405-3668
Fax 301-314-9281
silio@umd.edu

ENEE 350 Homework Problem Set 5

(Due: Class 11, Wed., June 18, 2008)

For these problems use the MAC-1 Instruction Repertoire that includes the HALT instruction and the corresponding microprogram that fetches, decodes and executes these MAC-1 instructions.

1. Compute the total fetch, decode and execution time in microinstructions for each MAC-1 instruction running on the Mic-1.

2. Consider a possible revision of the Mic-1 in which the two sequencing fields, COND and ADDR, are removed, giving 22-bit microinstructions. This will require the use of encoded microinstruction opcodes that include a microprogram jump instruction that obtains its target address from a combination of say the A and B fields. Don't worry about how the opcode encoding is done. The ALU N and Z bits are also latched, to make them available to a subsequent microjump. Compare the size of the original Mic-1 microprogram with the one needed for this new design.

3. What are the similarities and differences in the functions performed by the following two Mic-1 microinstructions? Explain.
   a:=a+a; if n then goto 0;
   a:=lshift(a); if n then goto 0;

4. Your only listing of a key Mic-1 microprogram was accidentally put into the automatic document feeder for the paper shredder instead of the photocopies next to it. You will now have to reconstruct the source program from a core dump. In the course of disassembly you come across the binary number

   11001000000100011001000000001000

   Rewrite it in microassembly language (MAL).

5. If main memory reads and writes took three microinstructions instead of two on the Mic-1, would the microprogram size be affected, and if so, by how much?

6. How many bit patterns are there for the following Mic-1 microinstruction?

   ac:=mbr+a; if n then goto 100;

7. The manufacturer of the MAC-1 has just received a telegram from its most important customer. The telegram reads: "Must have new instruction to shift AC left n bits, where n is low-order 4 bits of instruction. Need tomorrow morning 8 a.m. Send new microprogram by telegram. Hurry." Your job is to write the (MAL) microcode for executing this shift instruction, assuming that a co-worker of yours will take care of assigning an opcode and modifying the opcode decoding statements in the existing microprogram so that execution starts at control store address 85 (decimal).

8. Implement execution of the new MAC-1 machine instruction: **SWPB**

   Assume that the execution sequence starts in address 160 of control store and that all control store locations > 160 are free and available for your use. SWPB stands for "swap bytes" in which the left and right half bytes in the 16-bit ac register are swapped (i.e., exchanged) so that ac_{7-0}:= ac_{15-8} and ac_{15-8}:= ac_{7-0}. (Hint: Consider using the 8 ones in the SMASK register for the shift count instead of initializing a register to 8 and then decrementing it upon each shift of the ac or other registers.)

(another problem on page 2)

9. Print out tools.ps in the Documentation subdirectory of the class website and read the assembler and linking loader, and simulator user's manuals. Then do:

Programming Assignment Zero: For the program from Homework Set No. 1, called **prog0**, that you have already edited into a file do the following:

```
tap ee350
assem prog0
qpr -q <your favorite printer> prog0.list
qpr -q <your favorite printer> prog0.rel
load prog0
qpr -q <your favorite printer> prog0.abs
sim prog0.abs $EE350/halt $EE350/halt.pascal
  create prog0.mem_start and prog0.mem_end and then edit out excess zeros after
  the second line in each file, combine them and name them, and then use enscript to
  reduce font size and print them in landscape form; call the output file
  "prog0dump.mem"
qpr -q <your favorite printer> prog0dump.mem
tsim prog0.abs $EE350/halt $EE350/halt.pascal
```

Note: you can learn how to use enscript by typing "man enscript" (without the quotes); however, you can use the command

```
"enscript -r -fCourier6  -P<your favorite printer>  prog0dump.mem"
or (if 8 point font will fit on the page)
"enscript -r -fCourier8  -P<your favorite printer>  prog0dump.mem"
```

I create an alias in my home directory in the ".aliases" file called "lp" for "landscape print" so that I can simply type "lp prog0dump.mem" to generate the output on my favorite printer. You might consider doing likewise. Here's what the line in my ".aliases" file looks like:

```
alias lp "enscript -r -fCourier6  -P<my favorite printer>"
```

Note that in the alias statement you need the double quotes to cause the operating system to substitute the entire string of symbols for lp rather than stopping at the first space following the word enscript.

Note: for those of you coming into the Glue workstation from a remote Unix system, say from machine "mymachine.faraway.edu" which for the sake of example has IP address "123.45.67.890" Before ssh-ing to glue.umd.edu you should first issue the following two commands to mymachine: xhost + y.glue.umd.edu and xhost + z.glue.umd.edu (Machines y and z on the glue subnetwork are the current two server machines.) Alternatively, you could simply type xhost + which would permit any network machine to write to your screen.

Actually, you may need to use "ssh engr.dialup.umd.edu" (without the quotes) to log into one of the glue Unix workstations. In fact, engr.dialup.umd.edu is the preferred choice over glue.umd.edu because your login will then be assigned to one of the glue workstations that is least loaded with other work. Using the secure shell login (ssh) should pass the DISPLAY environment variable correctly. An indication that the DISPLAY environment variable is properly set is when another window pops up on your screen stating that there are glue news announcements to be read). Once you know the name of the glue machine onto which you have logged (not necessarily "y" or "z"), you may still need to open a local xterm window and issue the "xhost +" command for that named machine in order for your machine to accept and display X-Windows commands from it.

```
If the following occurs,
     ``ssh engr.dialup.umd.edu
     @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
     @       WARNING: POSSIBLE DNS SPOOFING DETECTED!          @
     @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
     The host key for engr.dialup.umd.edu has changed,
     and the key for the according IP address 129.2.98.133
     is unknown. This could either mean that
     DNS SPOOFING is happening or the IP address for the host
     and its host key have changed at the same time ...''
```

```
then before attempting to log in again, edit the file:
     ~/.ssh/known_hosts
in your home directory and delete the line starting with
     ``engr.dialup.umd.edu ...''
```

The problem is caused by ssh placing into your "known_hosts" file along with a key of some sort a line starting with "engr.dialup.umd.edu" and the IP address of the machine to which your login is directed. The next time you log in by typing "ssh engr.dialup.umd.edu" you may be directed to a different machine with a different IP address (and for that matter a different name); however, all ssh sees in the known_hosts file is the name "engr.dialup.umd.edu" and the IP address and key no longer match, and so warns you and rejects your attempt to log_in. The easiest way to fix this problem is to delete the line entered in the known_hosts file by your previous login.

If after logging into one of the glue machines and finding that the DISPLAY environment variable is not properly set (such as, if you could login using telnet), issue the following command to the glue machine:

```
setenv DISPLAY mymachine.faraway.edu:0.0     or alternatively:
setenv DISPLAY 123.45.67.890:0.0
```

(the colon zero.zero is your console display). You can check that the glue machine is sending X-Windows commands to your screen by typing: echo $DISPLAY and echo $REMOTEHOST These steps may be needed to get the sim simulator to display X-windows graphic screens on your remote machine. However, by using ssh you shouldn't need to do this setting of the DISPLAY environment variable (see the manual entry for ssh by typing "man ssh").

Since you can't send output to a glue printer, you will need to do

```
enscript -r -fCourier8 prog0dump.mem > prog0dump.mem.ps
```

or

```
enscript -r -fCourier8 -pprog0dump.mem.ps  prog0dump.mem
```

Use psFTP, or even better, WinSCP to transfer the ascii ".ps" file to your local machine and then print it there (possibly using ghostscript to do so). Before transfering the file you could alternately type

```
ps2pdf prog0dump.mem.ps prog0dump.mem.pdf
```

and then transfer the binary ".pdf" file to your local machine where you could use Adobe Acrobat to view and print it.

Note: psftp simply transfers the characters in the file; so if you want to print a text file without using either postscript or Adobe acrobat for pdf files on your local (IBM like) PC, you first need to use (on the glue workstation) the command unix2dos.

```
unix2dos -ascii <source_file> <destination_file>
```

and then transfer "destination_file" which will now have each line_feed character followed by a carriage_return. Going the other way, first use psftp to transfer the file to GLUE and then use dos2unix to convert the file by removing the carriage_return characters that follow line feeds. **BE CAREFUL** before using either unix2dos or dos2unix to read the manual page entry because you can lose (or corrupt) your source file if you don't watch out. For instance, if only the source file is named on the call, it is overwritten by the destination file; so it is a good idea to give the destination file a different name (or at least a different suffix).

It should be possible to avoid doing this insertion and deletion of carriage returns by using WinSCP.