CSE 320 Honors Project
Milestone #8

Milestone Deliverables

Use the "handin" system to submit the following file:

```
machine.c
```

I recommend completing the milestone by Tuesday, April 19 (it must be completed no later than Tuesday, April 26).

Please note that the "handin" system is expecting you to use "320H" as your course and "9" as your section.

Milestone Overview

The ARC+ microprocessor recognizes 40 different machine language instructions (a subset of those recognized by the SPARC microprocessor).

This milestone focuses on the processing of the CALL and JMPL instructions.

Milestone Resources

The SPARC Architecture Manual specifies the format of the instructions. In particular, the diagrams on pages 44-46 are relevant.

Milestone Preparation

The following files are available under "/user/cse320/Honors/Milestone08":

```
clock.h      -- interface to CLOCK
clock.o      -- implementation of CLOCK
registers.h  -- interface to REG, NPC, PC, IR and NZVC
registers.o  -- implementation of REG, NPC, PC, IR and NZVC
memory.h     -- interface to IMEM and DMEM
memory.o     -- implementation of IMEM and DMEM
components.h -- interface to DECODE and ALU
components.o -- implementation of DECODE and ALU

machine.c    -- implementation of the ARC machine (incomplete)
machine.make -- makefile for the ARC machine

test.s       -- assembly language file
```

I recommend the following steps:

a) Create a directory for this milestone and copy the following files:

```
/user/cse320/Honors/Milestone08/machine*
/user/cse320/Honors/Milestone08/test*
```

b) Use "make -f machine.make" to generate the executable named "arc".

c) Use the following command to generate a sequence of machine language instructions that will serve as input to the simulated machine:

```
/user/cse320/bin/arc_prep test
cp test testcases
```

Milestone Specifications

The incomplete version of the simulated machine ("machine.c") correctly handles the fetch phase for data manipulation instructions. You will extend the machine to handle the execute phase for data manipulation instructions, data movement instructions, SETHI instructions, CALL instructions, and JMPL instructions.

You will create the necessary signals and use the following components to execute the specified instructions:

```
DECODE -- instructor-supplied Decode Unit
REG    -- instructor-supplied Register File
ALU    -- instructor-supplied ALU
DMEM   -- instructor-supplied Data Memory
```

In addition, you may need other components to complete the processing. For example, you will use a multiplexer to handle the selection of the value to be written into the register file ("MUX-B" on the diagram). Also, you will use a multiplexer to handle the selection of the value to be written into the NPC ("MUX-C" on the diagram).

Since the ARC+ does not have windowed general-purpose registers, the following conventions will be used for function linkage and communication:

a) The calling function saves the PC in R15 (via the CALL instruction).

b) The called function assumes the return address is 8 bytes beyond the address in R15.

c) The address of the top of the current stack frame is in R14.

d) Up to six arguments may be passed to a function using R8-R13.

e) If appropriate, the return value is placed in R8.

When execution begins, the top-level function ("main") should initialize the run-time stack by placing the value 0x00002000 in R14.

At the entry point for a non-leaf function, it should reserve a new stack frame by decrementing R14 by the number of bytes needed for the stack frame. It should then save R15 (and possibly other registers) in the current stack frame.

At the exit point for a non-leaf function, it should restore the value of R15 (and possibly other registers) from the current stack frame. It should then discard the current stack frame.