Homework 9: SAM

Homework is due in class on the date indicated above. Late homeworks will be penalized by weighting errors on those problems that were completed past the due date by the number of days past the due date plus an extra 5% per day. Please indicate which problems, if any, took extra time.

For this homework, you are going to write a sequential description of a microprocessor and decompose this to a collection of communicating processes. You will be required to use the `chpsim` simulator to test the initial and final descriptions of the microprocessor.

1 Sequential CHP description of a SAM

25%

Write a sequential description of a processor that implements the Simple Asynchronous Microprocessor(SAM) architecture. Use arrays, named *imem* and *dmem* to respectively implement the instruction and data memories. You may assume that the *imem* is externally initialized.

It will be useful to write the sequential CHP in such a manner that it is possible to distinguish the part that computes the program counter, *pc*, from the part that executes the instructions. Name this process SAM. Simulate the SAM process in `chpsim` to verify that it is correct.

2 Isolating the memories and register file

25%

Since the implementation of a memory differs from the implementation of other logic, it is useful to isolate memories to their own process. Decompose your description from the previous part into 3 process such that:

$$SAM \equiv SAM\_2 \parallel IMEM \parallel DMEM$$

where $IMEM$ and $DMEM$ are processes whose interfaces only permit reads and/or writes to the respective memories. Do not remove the original CHP body of the SAM process from your code, simply add a META body to the process.

For similar reasons, it will be helpful to isolate the register file into a process that permits only reads and writes to one or more registers. Let this process, $REGFILE$ be such that

$$SAM\_2 \equiv SAM\_3 \parallel REGFILE$$

3 Isolating the fetch

25%

Often, the throughput of a microprocessor is limited by the rate at which instructions can be fetched from the instruction memory. Decompose the SAM_3 process into two process $FETCH$ and $EXEC$. Let the $FETCH$ process handle all accesses to the $IMEM$ and any manipulations of the *pc*. The $EXEC$ process implements all instructions that do not modify the *pc*.

$$SAM\_3 \equiv FETCH \parallel EXEC$$

4 Completing the decomposition

25%

Decompose the $EXEC$ process so that

- there is a process that handles all ALU instructions
- there is a process that handles all $DMEM$ instructions
- there is a process that handles all $SHIFT$ instructions

Decompose the $FETCH$ into 3 processes as follows:

$$\mathit{FETCH} \equiv \mathit{PCUNIT} \parallel \mathit{YMODE} \parallel \mathit{DISPATCH}$$

Let the $PCUNIT$ process handle all manipulations of the pc. Let the $YMODE$ process compute the y operand of the instruction and forward it to any processes that need it. The $DISPATCH$ process will forward the various fields of an instruction to the appropriate processes.

Simulate each level of decomposition in `chpsim` to convince yourself that no errors have been introduced during the decomposition.

Important: When you decompose a process, you should not create a second definition for the same process. Leave the chp body in place and add a meta body to the process definition. Chpsim will ignore the old chp body and execute the meta body, but the chp will remain in place for documentation/debugging purposes. As a result, you should **not** be handing in a seperate file for each part of this assignment.