UIC – ECE 366: Computer Organization II - Fall 2018

Project 2: ISA Design

For this project, you will design your own *ISA (Instruction Set Architecture)* with its relevant software and hardware packages.

This will be a processor with 8-bit instructions (including 1 parity bit) and 16-bit data (registers and memory), which you will optimize for two programs (ME - "Modular Exponentiation", and BMC - "Best Match Count").

Your ISA design should be very clear with the following:

- a unique name
- which instructions to support, and how to encode each within 7 bits
- number of registers, general-purpose or specialized
- memory addressing modes: How are addresses of memory constructed / calculated? (This applies to both data and instruction memory accessing: load/store, and branch instructions).

General specifications:

- Instructions and data are stored in a separate memory hardware (both begin at address 0):
- Instruction memory: 8-bit byte addressable, PC is initialized (automatically) at 0. This memory should be as large as you need to hold one program. You can assume that each of the two programs will be separately loaded in, so each will start at location 0 of instruction memory.
- Data memory: 16-bit double-byte addressable with the following access limit: for any instruction, it can either load from or store to (but not both) the data memory exactly 16 bits. The data memory should be able to hold at least 103 16-bit data entries. at least 7 bits
- When implemented, this will be a single-cycle cpu, so you cannot assume to have a general "multiply" or "divide" instruction. If you want to have some special, "powerful" instructions, you need to show its implementation in ALU as combinational logic, not sequential ones.

Optimization goals:

1. High Speed in SW: (i.e., minimizing dynamic instruction count for one or both programs).
2. Low Cost in HW: (i.e., making your CPU's hardware design simplified).

[**Program 1: ME - “Modular Exponentiation”**]

Compute the result (R) for: $R = 6^P \% Q$.

Input: the 16-bit positive numbers P and Q are initially stored in data memory:
P = Mem[0]
Q = Mem[1]

Output: the 16-bit answer *R* should be written back into memory address 2 by your program:
Mem[2] = R

Levels of completion:

1) (80% grade) Assume a fixed Q = 17.
2) (100% grade) Support general 16-bit P and Q.

[**Program 2: BMC - “Best Match Count”t**]

Find out the Best Match score (S = [0, 16] where 16 indicates total match) and count (C: number of best matches) from an array (Pattern_Array[]) of up to one hundred 16-bit patterns, to a given target pattern (T).

Input:

The 16-bit pattern T is initially stored in data memory address 3:
T = Mem[3]

The array begins at data memory address 8:
Pattern_Array[1] = Mem[8]
Pattern_Array[2] = Mem[9]
Pattern_Array[3] = Mem[10]
…

Output: the 16-bit answers S and C should be written back into memory by your program:
Mem[4] = S
Mem[5] = C

Levels of completion:

1) (80% grade) Only need to support “total matching”: S=0 and C=0 if no total matching is found.
2) (100% grade) Support in general any best-matching score