ICOM 4215 Project 1 Fall 2011 Processor Simulator

Today: March 7, 201

Due date: March 23, 2012

Points: 100 points (Penalties: Next day: -10 points, Two days late: -25 points, Three days late: -40 points, Four days late or more: not accepted)

Group project – Three students per group

Submission:

- Via oral exam, aka "Happy Hour".
- Report, via email, subject on the email: Project 1 ICOM 4215, students will lose 5 points if the subject is changed. Send the email to nayda.santiago@ece.uprm.edu. Email due time, 11:59pm.

Project

In order to understand how a processor works and the different aspects taken into consideration when designing a processor, we are requesting that you design a simulator for a simple processor. The following sections describe the processor.

General Processor Description: RISC AR4

The RISC AR4 is a processor designed by your professor, taking the ideas from the Simple Risc Processor, from the Jordan and Heuring textbook, a processor designed by Manuel Jimenez, Sunil Vaidya, Bradley Vansant, and Dave Dorner for the EE 813 graduate course at Michigan State University, and the processor designed by Adem Kader and Mustafa Paksoy for the E25 : COMPUTER ARCHITECTURE course at Swathmore University.

Processor Features:

- 8-bit internal data bus
- Internal 256-word 8 bit wide program memory
- 8 byte register file
- On chip 4 bits hardware multiplier/adder providing 8 bit results.
- 2 external I/O pins
- RISC instruction set: 21 instructions
    - 6 arithmetic
    - 4 logical
    - 5 data transfer
    - 5 control flow instructions
    - 1 machine control

Processor Block Diagram

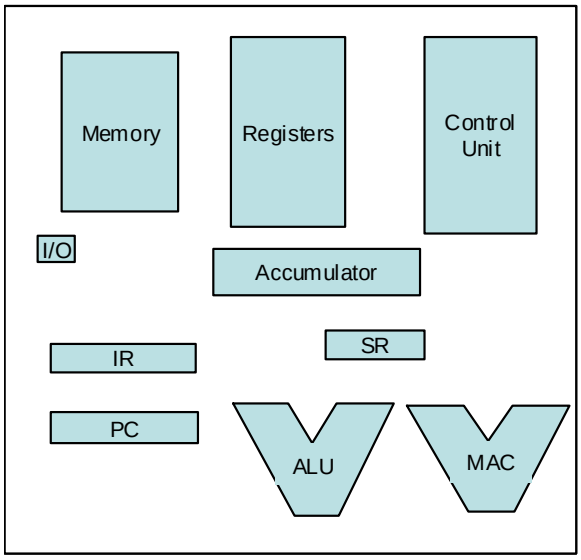

Figure 1: Block diagram of the RISC AR4

Memory and Registers

The size of the memory is 256 organized as 256 addresses of 1 byte each.

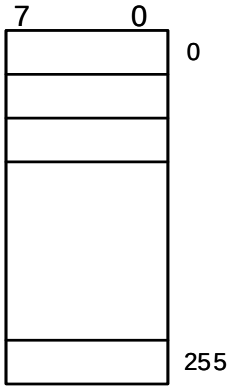

Figure 2: Visual illustration of the memory of the RISC AR4

Internally, the processor has 8 general purpose registers, 8 bits each. The names of the registers are from **R0** to **R7**.

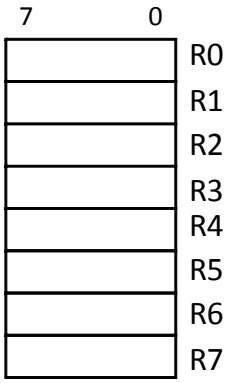

Figure 3: Visual illustration of the general purpose register structure of the RISC AR4

The processor has a 8 bit program counter called **PC**, an 8 bit accumulator called **A**, and a 16 bit instruction register called **IR**. There is a 4-bit status register called **SR**. The format of the Status Register is 

where Z is zero, C is Carry, N is negative, and O is overflow. When instructions are saved into memory, big endian ordering is used (A big-endian machine stores the most significant byte first).

Four addressing modes are supported by the processor:

a) Implicit
b) Immediate
c) Direct
d) Register indirect

The list below shows the different addressing modes supported and the corresponding instruction formats for each (see figures 4 to 7).

(a) *Implicit addressing*: The only operand needed is contained in the accumulator (A)

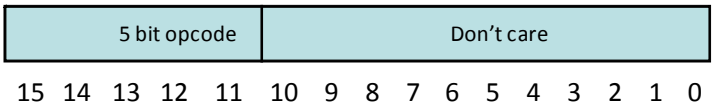

Figure 4: Instruction format for the *Implicit* addressing mode

(b) *Immediate addressing*: The data to be operated is part of the instruction itself.

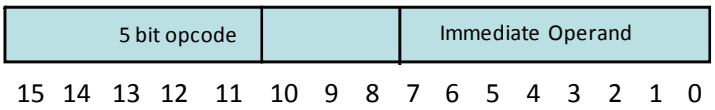

Figure 5: Instruction format for the *Immediate* addressing mode

(c) *Direct*: The memory location is indicated within the instruction.

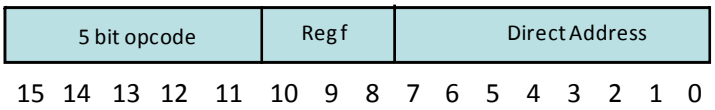

Figure 6: Instruction format for the *Direct* addressing mode

(d) *Register direct addressing*: Register f contains the operand.

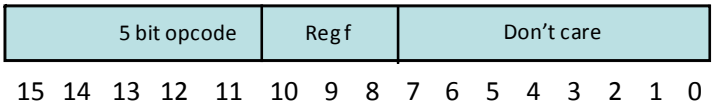

Figure 7: Instruction format for the *Register Direct* addressing mode

Instruction Set

The following table is a summary of the instruction set of the RISC AR4. Note that register f refers to one of the eight general purpose registers.

Table 1: Instruction set of the RISC AR3

Item	Opcode	Name (Mnemonic)	Operands	Addressing Modes	Operation	Details
1	00 000	AND rf	Accumulator, register f	Register Direct	A ← A and rf	Logical AND
2	00 001	OR rf	Accumulator, register f	Register Direct	A ← A or rf	Logical OR
2	00 010	XOR rf	Accumulator, register f	Register Direct	A ← A xor rf	Logical XOR
3	00 011	ADDC rf	Accumulator, register f	Register Direct	A ← A + (rf) +carry	Addition with carry
4	00 100	SUB rf	Accumulator, register f	Register Direct	A ← A - (rf)	Subtraction
5	00 101	MAC rf	Four least significant bits of accumulator , four least significant bits of register f	Register Direct	A ← A *(rf) + (rf)	Multiply accumulate
6	00 110	NEG	Accumulator	Implicit	A ← ^(A)	Two's complement
7	00 111	NOT	Accumulator	Implicit	A ← not(A)	Negate
8	01 000	RLC	Accumulator	Implicit	A ← A6..A0 &Cf, Cf ←A7	Rotate left through carry
9	01 001	RRC	Accumulator	Implicit	A ← Cf & A7..A1, Cf ←A0	Rotate right through carry
10	01 010	LDA rf	Accumulator, register f	Register Direct	A ← (rf)	Load accumulator from register f
11	01 011	STA rf	Accumulator, register f	Register Direct	(rf)← A	Store accumulator to register f

12	01 100	LDA addr	Accumulator	Direct	A ← [addr]	Load accumulator from memory location addr
13	01 101	STA addr	Accumulator	Direct	[addr]← A	Store accumulator to memory location addr
14	01 110	LDI Immediate	Accumulator	Immediate	A ← Immediate	Load accumulator with immediate
15	10 000	BRZ	Status register	Implicit	If Z=1, PC ←r7	Branch if Zero
16	10 001	BRC	Status register	Implicit	If C=1, PC ←r7	Branch if Carry
17	10 010	BRN	Status register	Implicit	If N=1, PC ←r7	Branch if Negative
18	10 011	BRO	Status register	Implicit	If O=1, PC ←r7	Branch if Overflow
19	11 111	STOP	PC	Implicit		Stop execution
20	11 000	NOP		Implicit		No operation

Arithmetic instructions use a 2's complement representation for negative numbers. This format is also used to compute memory addresses when accessing memory. Register 7 is a special register that will be used for branching conditions.

Processor Configuration

The processor operates with instructions located in main memory from address 0 to address 127. When the processor starts, it will always do so from location 0 and 1.

The system will also have two devices connected to I/O ports using the following memory locations:
250-251: 16 bits, input from keyboard
252 – 255: ASCII display, each byte will represent one ascii character

The information coming from the keyboard will be entered at the keyboard of the computer running the simulation. The display will be presented at the computer screen of the computer running the simulation. The data entered in the keyboard will have the corresponding ASCII value and the data written to the display should have the ASCII character corresponding to the code. The following figure illustrates a possible configuration of the graphical user interface for the simulator you will design.

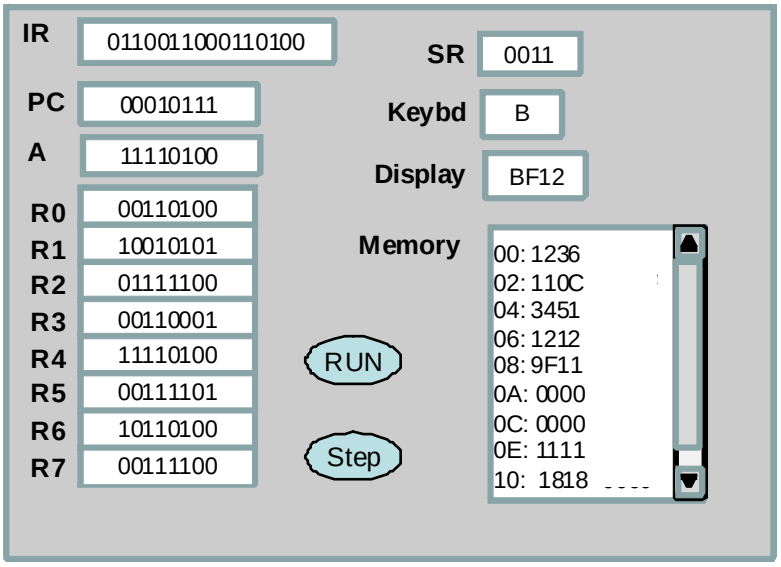

Figure 8: Possible graphical user interface arrangement

Project requirements

Simulator characteristics
The simulator must simulate all instructions in the instruction set, including all addressing modes. The simulator will run the instructions located in the main memory, starting with address zero (PC=0). The contents of memory are changed as a file with instructions is loaded into the simulator. Your professor will bring a simulation file on the oral exam day. This file will contain one line per instruction, and it will be represented in HEX characters (4 Hex characters).

The following example illustrates an input file:
7019
5900
70F4
5A00
7002
5B00
7028
5F00
6880
1900
F800

The input file may have any name and will be provided as input.

The simulator should run in two different modes: run or step. Run mode will allow programs to run from start to end. Step mode will run one instruction at a time. Please notice that the last instruction in any file should be stop. The simulator must show the contents of all registers, program counter, instruction register, and the contents of a section of memory. The directions in memory should be shown in HEX representation.

Projects that do not run in step mode will not be graded and a 0 Grade awarded to the project.

Your design may use a graphical user interface or a plain text interface. Use any programming language of your preference.