ROARANGE SYSTEM: USING CircuitMaker TO CREATE A SIMPLE COMPUTER

STUDENT PAPER

Christopher M. Lux
James E. Van Meer III
Wesley A. Hediger
Department of Mathematical Sciences
University of Northern Colorado
Greeley, Colorado 80639
ChrisLux@Juno.com
Zeddicus1@Juno.com
HED2084@Iname.com

ABSTRACT

For the required final project in Computer Architecture/Organization the authors selected creating a simple computer using the CircuitMaker software. CircuitMaker is a software simulation of electronic circuits. [Circ98] The simple computer we chose is the one described in M. Morris Mano=s book "Computer Architecture."[Mano93]

The instruction set and micro-operations for the computer are well described in Mano's book. These descriptions were followed in creating the computer. There are 8 registers and approximately 20 instructions including the input/output instructions.

This experience helped us understand how real computers work. In addition it can serve as a tool for students in later classes in helping to understand how the Mano computer is put together.

INTRODUCTION

The class CS 222 was based on computer architecture and digital logic. It covered from microcode to assembly language and its conversion and from simple computers to Pentium processing. Most of the teaching utilities were programs built by former students or professors like VM3 [Howe98], another simple computer with more instructions.

The Roarange computer implemented many of the ideas taught in the class. First the Roarange computer uses most of the instruction set created and described in the book by M. Morris Mano. [Mano93] The Roarange is a single instruction, single data or SISD

computer, meaning that the computer handles at one time, one word of data with one instruction. It uses a parallel bus transfer.

ABOUT MANO

The computer created by Mano was totally theoretical; it was only used for teaching purposes. Our contribution to this was to take his concepts and using the software "Circuit-Maker Pro 6.0", bring this theory to life.

The instruction set for the Roarange system can be broken up into three groups: Memory-reference, Register reference, and Input-Output.

The Memory-Reference instructions deal with data stored in memory. For example, if an AND instruction is used, it take the value in the accumulator and AND's it bit wise with the data in the specified memory location. Other instructions are: ADD, which adds the memory location to the accumulator. LDA loads a memory location into the accumulator. STA stores what is in the accumulator to the memory location. BUN unconditionally branches to a memory location. BSA will save the program counter value and branch to a memory location. ISZ increments the memory value and skips the next instruction if that result is zero.

Register reference instructions deal with the manipulation and testing of data inside registers. For example, if an INC is entered for an instruction, the computer takes the data inside the accumulator and adds one to it, then stores it back into the accumulator. Other instructions are as follows. CLA, clears the accumulator. CLE clears the e-flag, which is the overflow bit. CMA compliments the accumulator. CME compliments the e flag. CIR circulates the data in the accumulator to the right once. CIL circulates the data in the accumulator to the left once. INC increments the accumulator with one. SPA skips an instruction if the accumulator is positive. SNA skips an instruction if the accumulator is negative. SZA skips an instruction if the accumulator is zero. SZE skips an instruction if the e-flag is zero. HLT halts the program.

The third group of instructions is the basic Input-Output set. This set of instructions deals with the input and output operations. For example, if an INP is entered for an instruction, the computer takes the data from the input register and places it in the accumulator. Other instructions are as follows. OUT takes the value in the accumulator and outputs it to the output register. SKI skips the next instruction if the input interrupt flag is on. SKO skips the next instruction if the output interrupt flag is on. SKI and SKO can be used in a busy-wait loop sometimes called programmed I/O. Both ION and IOF were not implemented. They were used for implementing interrupt driven I/O.

An instruction uses a 16-bit 'word'. For a memory reference instruction the word is split up into three different pieces. The fifteenth bit is used to determine whether or not the code is using direct or indirect memory addressing. The fourteenth through twelfth bits are used to determine which instruction is to be executed. The eleventh through the zeroth bits are used for the address of the memory location to be accessed. The register instructions are recognized by a hexadecimal 7 in the first four bits. The remaining bits determine which register instruction is to be executed. The I/O instructions are recognized

by a hexadecimal F in the first four bits and the remaining bits control which I/O instruction it is.

ABOUT CIRCUIT MAKER

CircuitMaker is a virtual electronic workbench. It was developed to help professional electronic circuit developers test their ideas in a virtual environment. It has many features to help build electronic circuits. For one, the program is designed to simulate the drag-down effect to the current when multiple devices are connected to the output of another device. This is usually referred to as the fan-out. Also wires that need grounding must be grounded in order for the circuit to run properly. It also has a tracking feature to help detect errors within a circuit. The wires change color as the state of the wires change between on, off, and null states. CircuitMaker also provides a total cost as to how much it would cost to develop the circuit. It comes with a very large library of commonly used components. These include many different chips, resisters, switches, displays, as well as the basic building block logic gates. Macros are special components that look like a standard IC but that can be designed and programmed by the developer. This environment enabled us to quickly build, and test the necessary parts of a basic computer.

OUR PROJECT

One problem we encountered while building our system is the nanosecond differences in timing between using data and storing it into one of the registers. To solve this problem we chose to have the logic change on the leading edge of the clock pulse and the registers set the value on the trailing edge. Another problem is that our circuitry was suffering from major drag-down effects due to the fan-out problems. We had to strategically place buffers throughout the system to help provide the needed current through the wires to make the devices operate correctly. Also, in CircuitMaker, when resetting the circuit while lines were selecting a current memory location, it would delete the data at that memory location. To solve this we moved the HLT command from 7001, which would reset memory location 001 or the second instruction, to FFFF making it delete the very last memory location. We decided that this was okay because programs will be less likely to use this memory location. The RAM provided by CircuitMaker did not have enough bits to hold a full Mano Computer Word. The RAM memory locations used only two hex places while we need four. So we had to use four sets of two RAM chips simultaneously for each pair to simulate an entire Mano Computer Word.

CircuitMaker includes a bus which can be used to include many bus wires inside of a single cable. We used the bus to group the wires together so we could use less space. Basically we ran a bus wire around the computer to transfer bits simultaneously. To control which registers were to dump to the buses at what times in the program, we used two chips with eight three-stage buffers.

MANO INSTRUCTION SET

The following table contains all the instructions that were implemented for the project.

Roarange System Machine Code Instruction Set Closely Based on Mano [p 133]

Symbol	Hex Code		Description
	Dir. Addressing	Indirect Addressing	
AND	0xxx	8xxx	AND memory with word to AC
ADD	1xxx	9xxx	Add memory word to AC
LDA	2xxx	Axxx	Load memory word to AC
STA	3xxx	Bxxx	Store content of AC in memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and save return address
ISZ	6xxx	Exxx	Increment and skip if zero
CLA	7800		Clear AC
CLE	7400		Clear E
CMA	7200		Compliment AC
CME	7100		Compliment E
CIR	7080		Circulate right AC and E
CIL	7040		Circulate left AC and E
INC	7020		Increment AC
SPA	7010		Skip next instruction if AC positive
SNA	7008		Skip next instruction if AC negative
SZA	7004		Skip next instruction if AC zero
SZE	7002		Skip next instruction if E is 0
HLT	FFFF		Halt computer
INP	F800		Input character to AC
OUT	F400		Output character from AC
SKI	F200		Skip on input flag
SKO	F100		Skip on output flag

RESULTS

Using the architecture provided by Mano and some ideas of our own we were successful in completing the Roarange System. It has a full complement of instructions including: Input-Output, Memory Manipulation, and Register Manipulation.

We feel that we could have used fewer gates than what we ended up having. Instead of using four gates for data register, we could have used only two with better control over the data transfer. We would have probably used the bus wire throughout the entire system

instead of in only a few areas. The memory chips did not hold data when the file is not active, so we would have to reprogram the memory each time that we restart CircuitMaker. In short, we learned a lot from this project and gained experience. With this experience, if faced with this project again we would probably design and build this system very differently. This project did show us the power of CircuitMaker

We have learned a lot from this project. We learned that attaching multiple devices to the output of another device can drag down the output current and cause the devices to work incorrectly. A buffer will help maintain the correct current flow. Also we learned that not everything is done simultaneously with the clock pulse. It takes a longer time for some circuit components to settle to their final values.

BIBLIOGRAPHY

[Mano 93] M. Morris Mano, Computer System Architecture. Third Edition, Prentice Hall. New Jersey 1993.

[Circ98] CircuitMaker Version 5, http://www.microcode.com/ , 1993 , 1998.

[CircS98]Downloadable Free Student Version of Circuit Maker @ **http://www.microcode.com/student.htm**

[Howe98] A virtual computer written by Chuck Howerton found at: http://clem.mscd.edu/~howerton/VMx/VMx-F98.html