THE SIMULATION OF A MICROPROGRAMMABLE COMPUTER

Clement Luk*
and
Larry L. Wear
Department of Computer Science
California State University, Chico

Introduction

In this paper we present a simulator for the HP 21MX [1], a microprogrammable computer. This simulator was developed as an aid to teach microprogramming. The simulator, called C21MX, is written in both FORTRAN and assembly language. Assembly language is used for bit manipulations, and for routines that are straight forward. A routine is considered to be straight forward if it involves only simple operations that are mutually exclusive. This often happens for the functional simulation of a unit controlled by an encoded microinstruction field. Other routines are written in FORTRAN to take advantage of its I/O convenience and easy interface with assembly language routines.

Simulator Directives

C21MX operates under the direct control of a user sitting at a terminal. It is invoked with 5 optional parameters. Two of them set the upper limits for 256 words each of control memory and main memory. Due to physical memory limitations, we decided not to simulate the complete address space of control and main memories. Because the typical mode of operation in the student environment requires only a few words of main memory and less than a page (256 words) of control store, this does not appear to be a severe limitation.

Two other parameters set the microprogram counter (MPC) for the starting control memory address, and the program counter (P register) of the main memory program address. The last parameter sets the next breakpoint (BP) address (a control memory address).

The C21MX will execute microprograms until a microinstruction specified by BP has been executed. It then prints the BP address and the mnemonics of the microinstructions executed; it then returns control to the user for further directions. There are 4 ways to specify the next BP. The first is to use the optional parameter available when C21MX is initiated. The second is to use the breakpoint directive (BR,n) where n is the next BP address. The third and fourth methods are available in the single step mode in which case C21MX will return control to the user after the execution of each and every microinstruction. Single step mode can be entered using a BR,* directive. C21MX will integrate a system message location (currently bit 15 of the switch register on an HP 2100A host) after the execution of each microinstruction; this affords the user the ability to interrupt C21MX at any time. This is useful when a microprogram loop occurs.

Directives to display the contents of the registers, the main memory and the control memory are available. Display of a control memory location prints the microinstruction in both binary (octal) format and mnemonic format.

The contents of the registers, main memory and control memory can be modified with binary (octal) values. Control memory may also be modified by inserting the assembly language mneumonic for each field of the instruction.

Other directives available to allow the user to 1) terminate C21MX or to resume execution at the point of suspension or with a new MPC value, 2) load preassembled microprograms from a disk file or a paper tape, and 3) select an on-line printer for a record of simulator run or for voluminous output (trace).

The last directive discussed is trace, TE. Tracing refers to the printing of certain registers after each microinstruction. While its use is not recommended for normal programming, we believe it should be available. During peak usage of the simulator, there may not be enough terminals for on-line use to debug a programming project. The TE option allows the student to bring home a history of the simulated run for better understanding of the operations of the control unit and his microprogram.

A summary of the directives can be found in Appendix A.

Organization of C21MX

We shall present the simulator in terms of its control blocks, memory operations and general control. The information could be useful in the development of simulators of other microprogrammable computers.

Control Blocks

The simulator environment is specified in 10 control blocks. Control block 1 is used for major simulator control and maintenance. In this block are 1) controls for selective printing (dumping) of the control blocks should its maintenance so dictate, 2) break point information and 3) the clock time. Control block 2 contains all the hardware registers. Control block 3 contains simulator I/O unit information and special flipflops for simulated central I/O interrupt control, microinstruction "repeat" control, and register A register B access control. Control block 4 contains simulated hardware flipflops. Control block 5 contains information for simulated data paths. Control block 6 contains information affecting memory operations and multi-register operations. Control block 7 contains front panel registers. Control block 8 contains the different fields of encoded microorders. Control blocks 9 and 10 contain simulated control memory and main memory and their respective limits.

Memory Operations

The memory read/write affects the memory address register and the memory data register. The referenced address is checked for "out of bound" condition; if it should be out of bounds, a warning message is given. In this case no writing is allowed, though reading is not inhibited. The major problem with the simulation of

* Current address: CTRCC, Lawrence Livermore Laboratory, University of California, Livermore, CA 94550

memory read/write operations is timing, especially with main memory. Main memory control for C21MX involves a read indicator, a write indicator and a memory lock out. The read and write indicators are used to indicate the read or write operation desired and to ensure synchronization with the other functional units.

When the read (write) indicator is not zero, a main memory operation is required. The machine may go into a "freeze" during which most operations are disallowed. However, operations outside of the central processing unit (CPU) usually continue. After the memory operation has started, the read or write indicator will start "count down". When it is zero, data will be available for reading, or the memory data register will be latched for writing. Note that although read and write operations appear to be mutually exclusive, two indicators are required. In addition, when a memory operation is requested, it may not be allowed to begin at all and even the "freeze" may not start because memory may be busy. Memory operation for the last request may not be complete, or an external request (e.g. DMA) may steal a memory cycle. The proper memory simulation requires a memory lockout to either queue the access request or simply indicates that memory is not available.

General Control

The HP21MX microinstruction format can be found in Appendix B. Each instruction is separated into different fields called microorders to allow functional encoding within each field. For example, the ALU field of a type 1 instruction allows 32 ALU functions as encoded in 5 bits. This encoding is convenient as long as the functions are independent. In this case the operations can be simulated with one assembly language routine. The field is used as an index to a jump address table to different code blocks to perform the required functions.

Very briefly then, the general control is:

(1) Advance the master clock (this clock provides the basis for all timing control and synchronization)

(2) Check timing conflicts and synchronization

(3) Check main memory operations

(4) Check repeat current microinstruction option

(5) Check breakpoint option

(6) Fetch next microinstruction and separate the different fields. Increment MPC (microprogram counter).

(7) Determine the type of instruction and call in the controlling routine (there is one controlling routine for each type of instruction).

Concluding Remarks

The C21MX simulator was written as a teaching aid for microprogramming. It provides information on the state of the simulated computer at the end of a discrete time interval. It affords the student an opportunity to understand potentially parallel operations of the central processing unit. It also can be used as a debugging aid for microprogram development. However, the output of this simulator is still primitive. Improvements could be made to employ graphics media to show to data paths.

Appendix A

How To Use C21MX

C21MX (HP21MX simulator) may be called with parameters P1,P2,P3,P4,P5. These parameters are optional and their uses are:

(1) P1 - Set next breakpoint address at P1

(2) P2 - Set the last word address of simulated control memory (note that only 256 words of control memory is simulated)

(3) P3 - Set the last word address of simulated main memory (note that only 256 words of main memory are simulated)

(4) P4 - Starting control memory address (MPC)

(5) P5 - Starting main memory address (P)

P1 through P5, if present, are decimal numbers. The default values for these parameters if they are absent (or input as zero) are:

P1 - Simulator will suspend operation after each microinstruction, i.e. single step mode is entered

P2 - 767

P3 - 255

P4 - 512

P5 - 64

In addition, if bit 15 of switch register is set, the simulator will enter single step mode.

Simulator Directives

1. MM,n,v — Modify Main memory n with value v.
2. MC,n,v — Modify Control memory n with value v. If v is an *, then accept a microinstruction field by field.
3. DM,n — Display Main memory location n.
4. DC,n — Display Control memory location n
5. BR,n — BReakpoint: simulator will suspend operation after executing the instruction at location n. If n is an *, simulator will suspend operation after each microinstruction, i.e. single step mode is entered.
6. GO[,n] — GO, resume microprogram at location n, or at the point of suspension if n is absent.
7. KI — KIll execution.
8. LB,n,m — Load Binary file n into control memory starting from location m. Only 1 module (256) words may be loaded at one time. If, n=TAPE, then binary paper tape will be loaded.
9. TE,n — Trace Execution with option n. n represents a combination of the following option for the trace (printing) of different information.

n = 1	X,Y,A,B,RAR,RIR
n = 2	S1,S2,S3,S4,S5,S6
n = 4	S7,S8,S9,S10,S11,S12
n = 8	L,CNTR,OVFL,E,TBZ,SAVE
n = 16	FLAG,AAF,BAF,ONES,COUT,ALUQ,AL15
n = 32	IR,M,T,P,S,CIR

Simulator Directives
(continued)

For example, n = 3 will result in the printing of X,Y,A,B,RAR,RIR (for n = 1) and S1-S6 (for n = 2). The combination will depend on the binary representation of n.

10. LU,n — List Unit is set as n.

11. MR,n,v — Modify Register n with value v. n can be A,B,X,Y,M,T,L,P,C(CNTR),I(IR),U(RAR) or V(RIR).

12. DR,n — Display Register n. (see 11. MR,n,v for allowable value of n)

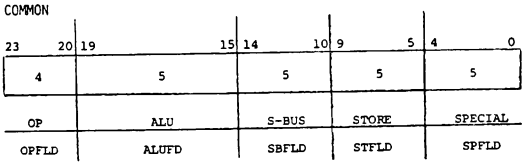
References

[1] Hewlett-Packard, Microprogramming 21MX Computers, Cupertino, California, 1974.

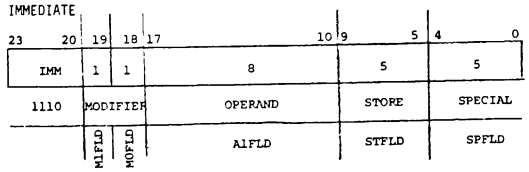
Appendix B

Instruction Format

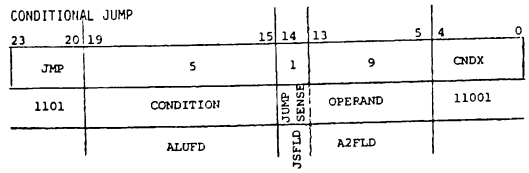
Type 1 COMMON



Type 2 IMMEDIATE



Type 3 CONDITIONAL JUMP



Type 4 UNCONDITIONAL JUMP