DESCRIPTION AND SIMULATION OF MICROCODE EXECUTION

Alice C. Parker Andrew Nagle
Department of Electrical Engineering
Carnegie-Mellon University
Pittsburgh, Pa. 15213

Abstract

This paper presents a facility for the description and simulation of the inner machine of microprogrammable processors. The use of this facility for teaching microprogramming and for research is discussed. Twelve simulations are described, and an example simulation shown. Observations about the research are presented, and future research outlined.

1. Introduction

This paper describes a preliminary investigation to analyze microprogram control structures by description and simulation at the register-transfer level. The original motivation for this was to provide participants in a graduate course in microprogramming with a facility for writing microcode for a variety of machines. We also wanted the class to gain an understanding of the underlying hardware which executed the microcode. In addition, it has recently been shown by Siewiorek and Snow [Snow 77] that much of the PDP-11 family performance variations are due to the control implementations. This motivates further the use of descriptions and simulations of inner machines for evaluation of the control hardware itself.

Section 2 of this paper describes the language and facility used for the descriptions and simulations. A summary of the machines described and simulated follows in Section 3, along with parts of an example simulation. Section 4 contains a brief discussion of the results of the simulations.

Special acknowledgements go to Mario Barbacci for supplying the original idea and to Bob Richardson, Al Dunlop, Mike Powell, Ted Elkind, Bill Paulsen, Steve McConnel, Richard Blum, Jim Crowley, William Wu, Doug Wiedemann, Konrad Lai, Hal Bellis and Lou Hafer who wrote and performed the original simulations.

2. The Language And Simulation Facility

The facility used to obtain the results presented here consists of a compiler and simulator. The user writes a behavioral description of the hardware to be simulated using the ISPL language [Siewiorek 74]. The description is compiled and the output from the compiler drives an interactive simulator. This simulator, used for the Computer Family Architecture evaluations [Barbacci 77], has most often been used for simulations of processors executing machine level instructions. These simulations, on the register-transfer level, contain implicit information about the control of the processors. For example, a sequence of statements like:

```
INSTR.REG ← M[PROG.CTR]
NEXT PROG.CTR←PROG.CTR + 1
```

in ISPL causes the instruction register to be loaded with the contents of the memory location pointed to by the program counter. Then, the program counter is incremented. The signals causing these operations to occur may be asynchronous or synchronous and may originate in a microstore, a programmable logic array or flip-flops. The control transfers and signals themselves are not simulated. Obviously, in order to evaluate the control architecture, it must be explicitly described and simulated. This is a new and powerful application of both the ISPL language and the simulator.

The ISPL simulator used by the class had the following capabilities:

- *The user can start the simulation at any procedure name in the ISPL description, can continue the simulation after a breakpoint (which he has set) and can exit the simulator
- *The user can collect satistical data about the use of variables/constants and the execution of labeled procedures and statements
- *The user can trace, set and interrogate values of the registers and memories declared in the ISP
- *The user can specify input and output files for access and storage by the simulator

Some of these capabilities are illustrated in Section 3 with an example.

3. The Simulation Experiment

Twelve different microprogrammable processors were described and simulated (See table 1). One of these, the convolution processor, was a paper design by one of the course participants. [Crowley 76]. Aspects of each processor architecture presented difficulties in the description itself and/or in the actual simulation. These will be discussed later.

The sophistication of the simulations varied widely. Some simulations executed a few lines of microcode to exercise the microcode and data paths. These will be referred to as TYPE A. Other simulations executed microprocedures which decoded and executed one or a few assembly level instructions. (TYPE B).

The most sophisticated simulations executed microcode which supported an entire target machine. Assembly language routines could be executed with this TYPE C simulation.

3.1 The Nanodata QM-1 machine was the subject of two separate simulations. The reason for this is that it has three levels in its control hierarchy (assembly language, microcontrol, and nanocontrol) instead of the usual two (assembly language and microcontrol). The micro-level control consists of a vertical microprogram which interprets the assembly language instructions. The nano-level control contains horizontal nanoinstructions that interpret the microinstructions. For this experiment, aspects of both the micro-level and the nano-level control were described and simulated. The microlevel description assumed an underlying nanocontrol but was written so that the effects of the nanooperations, not the execution of the nanocode itself, was simulated, much as the conventional ISPL processor descriptions assume an implicit underlying control. Even so, the description was lengthy in comparison to other descriptions.

For the microlevel simulation, microcode was written to fetch and execute machine level instructions

which load the memory with values contained in the local store. (TYPE B simulation). This microcode sequence exercised the data paths normally associated with the microcontrol and demonstrated the control hierarchy.

Although the QM-1 nanocontrol is straightforward, its description is also long. About a third of the nanoprimitives and most of the registers have been described with the ISPL language. The simulated nanoprogram fetches microinstructions from the control store and executes them. The two microinstructions executed are LD A, B which loads register pointed to by A with the contents of the store addressed by the contents of register B, and ADD A, B which adds registers selected by A and B (TYPE B simulation).

3.2 Selected portions of the INTERDATA 8/32 microprogram execution were described and simulated. The memory Access Controller (Relocation Registers), the Input/Output Unit, the Format ROM, the privileged ROM, the Interrupt Control, and some of the Arithmetic Logic Unit's functions were not implemented. Two microroutines were simulated - one which transferred data from one memory location to another and one which rotated a 16 bit field a variable number of times. The transfer routine moved seven words andthe rotate routine rotated the same field eleven different times, rotating the first time once, and the eleventh time eleven times. (TYPE A Simulation). The simulator executed a total of 6119 RTM* operations.

3.3 Simulating the DEC PDP-11/40 micro level machine proved to be difficult. This was due to two things. First, clock control is explicit in each microinstruction - clock functions are selectable. Second, interface to the asynchronous world over the UNIBUS™ occurs by stopping the clock and waiting for the bus transaction to be completed. The first feature could not be implemented since ISPL does not support a clock feature, and all operations in ISPL are assumed asynchronous. ISPL does allow description of asynchronous operations with the WAIT and DELAY statements, but the simulator did not support these. Unfortunately, two other features are still missing from the PDP-11 micro-level description at present - the complex microcode branching and some ALU control. In addition, the data display, past microprogram counter and microprogram interrupt have been omitted. However, there is enough of the machine description so that a simulation program to execute an 11/40 machine level ADD instruction was written and executed. (TYPE B simulation).

3.4 The Data General ECLIPSE, with its straightforward microcode and hardware, was less difficult to describe and simulate. Virtually all of the inner machine behavior except for the I/O channel was described in the microcode. A simple simulation test program, written in the ECLIPSE machine language, adds together a list of numbers, terminating when a negative number is reached (TPYE C simulation).

3.5 Virtually all the HP21MX processor micro machine was simulated. The 21MX executes vertical microcode with most parallel activity in the I/O. Some of the ROM table lookup of microprogram addresses was omitted due to lack of information about the specifics of this mechanism. In addition, a few microinstructions were omitted due to lack of information about the operations they were to evoke. The main body of the inner machine description is shown in Figure 1, along with definitions of the variables. The processor begins each new cycle of microinstruction execution by testing for CSAR=0 and halt or interrupt. If neither has occurred, the microinstruction is fetched and the microaddress incremented. The repeat bit is used for sequential mathematical calculations (multiply and divide, for example). The MXEQ routine, which executes the opfield of the microinstruction, along with routines which execute the other fields, are found in the declaration section of the ISPL description of the processor. Figure 2 contains a block diagram of the processor, and the instruction formats are found in Figure 3. The ISPL description consists of twelve procedures, each of which describes some portion of the micro machine behavior. The control store was loaded with the manufacturer supplied microcode* and a short assembly language program was written to add $M[105_8]$ to $M[107_8]$ and store the results in $M[10007_8]$. (TYPE C simulation). Some of the interactive simulator commands and the simulator output are shown in Figure 4. This simulation required 3248 RTM operations.

A second microprogram was written to multiply the A register by scratchpad register 01 and store the results in the accmulator. This program executed 1370 RTM operations.

3.6 Since the INTEL 3000 series of chips can be configured in a variety of ways, the ISPL description of these chips was written with the central processing element (CPE) and the microprogram control unit (MCU) as procedures. The reference used for this is [INTEL]. The main body of the ISPL program contains only seven statements which evoke the operations and procedures. Thus, reconfiguring the system to be simulated involves adding/deleting procedures and changing the main program. The configuration used for this simulation had no assembly language instruction fetch/execute capability; programming of the system was assumed to be on the microcode level only. The simulation used exercised the data paths of the CPE and MCU without emulating another architecture. (TYPE A simulation).

3.7 The Digital Scientific META micro machine was described completely except for a small amount of timing, I/O, and other logic. Two major references were used in writing the ISPL description [DIGITAL SCIENTIFIC 73] The simulated META 4 was microcoded to emulate an IBM 1130 (TYPE C simulation). Two IBM 1130 assembly level programs were executed - one multiplied a negative number by a positive number, and the other loaded the accumulator and performed an addition. The multiply routine executed 8340 RTM operations, while the load and add routine took 2661 operations.

3.8 The BURROUGHS B1700 micro-control description was large. A few parts of the B1700 controller were omitted - parts of the register file, the cassette control, and the "overlay M string" operation, for example.

The simulations performed were a 24 bit multiply and a bubble sort written in B1700 microcode (TYPE A simulation). The multiply simulation executed 3281 RTM operations, much more than the 21MX multiply and less than the META 4 multiply (which was emulating an 1130). Source for the B1700 microcode is [BURROUGHS 72].

*RTM operations are operations compiled from the ISPL description. These could be arithmetic operations, register-transfers, branch merges, joins of parallel paths, etc. The number of RTM operations required to execute a given microprogram is a somewhat rough estimate of the complexity of the micro-machine. It cannot be considered an accurate measure since it is sensitive to the level at which the ISPL description is aimed.

*A good test of the accuracy of the ISPL description.

3.9 The ISPL description of the MICRODATA 1600 micro machine contains the commands and registers available on the basic machine. Two simulations were performed, both of which computed the Fibonacci numbers. One transferred data by using the XOR function and one used the T register. The first took 4881 RTM operations and the second 7975.

3.10 The convolution processor described with ISPL was a hypothetical architecture. A small two-dimensional array of values was convolved with another array in the simulation program. The processor is a pipelined structure which is capable of performing the convolution calculation, reading a data point and calculating the address of the next image point, all in parallel [CROWLEY 76].

All of the above descriptions with the exception of the convolution processor are being or will be inspected by the appropriate machine users or designers to determine accuracy of the ISPL descriptions. Observations made in the process of writing these descriptions and running the simulations are discussed next.

4. Discussion

The goals established at the onset of the experiment were (1) to determine the utility of ISPL for describing and simulating micro machine architectures and (2) to compare the micro machines of several microprogrammable processors using an ISPL simulator as a measuring tool. ISPL had been used previously to describe and simulate the instruction set level of processors; this was its first application to micro machines.

4.1 Class participants reported that ISPL was easy to learn and use, and that its application to micro machine description seemed natural. The behavioral nature of the language made it possible to omit low level details (such as read or write signals, ALU function select signals, and bus protocols) and write the description in terms of register transfers. The major weaknesses of ISPL revolved around its inability to express timing considerations, and the fact that buses and I/O lines appear the same as registers in the descriptions and simulations.

These observations meet the first goal of the experiment, and lead to the following assertions:

*ISPL is useful for describing micro-machines

*The fact that manufactures supplied microcode can be used in simulation establishes the accuracy of a description.

*The ISPL language and simulator could be applied during the design of a processor to describe the micro machine, develop the microcode, and verify its performance by high-level simulation.

*Incorporation in the simulator of a facility to permit timing of RT operations and simulation of parallel activity would improve the realism of the simulations.

4.2 Interpretation of the results of the simulations to compare micro machine architectures is not yet possible. Since a different person wrote each description, there is a wide variation in the description "style" and the level at which each description is aimed. Therefore, if one simulation executes more RTM operations then another, it could be because (1) one processor is more complex than another, (2) one processor description is more concise than another, (3) one simulation algorithm is more complex than another, or any number of causes. No definite conclusions can be drawn until the descriptions and test programs are standardized.

4.3 Since the experiment, some progress has been made in the development of the ISP language. A successor to ISPL, called ISPS, now has a compiler, and an upgraded version of the ISPL simulator supports it. The upgraded simulator includes a limited timing facility and will simulate parallel activity. There are as yet no emperical results to show how useful these facilities are for analysis purposes.

4.4 Future research into micro-architecture evaluation will involve standardizing the ISPL descriptions of the micro machines and simulating each machine with a test program which performs functions identical to test programs executed on the other machines. The real value, however, of this type of simulation appears to be in allowing changes to the control architecture by manipulating the ISPL description. These effects of changes could then be measured with simulations.

4.5 Micro-level descriptions and simulations have been shown to be feasible and straightforward. They prove to be effective tools for microcode debugging, microde testing and documentation. In addition, formal verification and automated generation of microcode could be done based on an ISPL description of the microcontrol.

References

BARBACCI 77 Barbacci, M., et al., "Architecture Research Facility: ISP Descriptions, Simulation, and Data Collection", AFIPS Conference Proceedings, 1977, NCC, AFIPS Press, 1977.

BURROUGHS 72 Burroughs B1700 Systems Reference Manual, Burroughs Corproation, Detroit, 1972.

CROWLEY 76 Crowley, James, "The Design and ISPL Simulation of a Micro-Program Controlled Convolution Processor", Department of Electrical Engineering, Carnegie-Mellon University, 1976.

DIGITAL 73 META 4 Computer System Hardware Description, Publication No. 7050M0, and Computer System Microprogramming Reference Manual, Publication No. 7043M0 Corporation, 1973.

HEWLETT 74 Microprogramming 21MX Computers - Operating and Reference Manual, Hewlett-Packard Company, Cupertino, Ca., 1974.

INTEL INTEL Series 3000 Reference Manual, INTEL Corporation, Santa Clara, Ca.

SNOW 77 Snow, Edward, and Daniel Siewiorek, "Impacts of Implementaiton Design Tradeoffs on Performance: The PDP-11, A Case Study," Department of Computer Science and Electrical Engineering, Carnegie-Mellon University, 1977.

This research was supported by the U.S. Army Research Office under grant #DAAG29-76-G-0224, and the National Science Foundation under grant #GJ-32758X.

MANUFACTURER	PROCESSOR	PROJECT STATUS / PORTIONS SIMULATED
NANODATA	QM-1 microlevel processor	ISPL description assumes underlying nanocontrol, simulation exercised microcontrol logic and data paths by running small microprograms
NANODATA	QM-1 nanolevel processor	One third of nanoprimitives implemented small test nanoprograms run to fetch and execute microinstructions
IBM	360/67	Incomplete ISPL description
INTERDATA	8/32	Partially complete ISPL description, small test microprograms run
DIGITAL EQPT.CORP.	PDP-11/40	Incomplete description, current code debugged, test microprograms run
DATA GENERAL	ECLIPSE	Mostly complete except for I/O, runs test assembly language supported by the microcode
HEWLETT-PACKARD	21MX	Mostly complete, running test assembly language supported by the microcode
INTEL	3000	All microoperations implemented for a simple MCU-CPE configuration
DIGITAL SCIENTIFIC	META 4	Debugged mostly complete ISPL, microcoded to emulate the IBM 1130 instruction set, runs 1130 programs
BURROUGHS	B1700	Most microoperations implemented, test microprograms run
MICRODATA	1600	Basic 1600 completely described, test microprograms run
	Convolution Processor	Paper design, runs a test microprogram

Table 1 Microprogrammable Processors Described and Simulated

```
[011] !this is the main body of the simulator for the HP21MX
[011] !  it fetches microinstructions and executes them and
[011] !  fetches the next microinstruction ...
[011] !
[011] CSAR←0    NEXT                !start off at microlocation 0000
[011] MLOOP:=(DECODE REPEAT =>
[012]   !
[012]   !normal sequential execution
[012]   !
[012]   ((IF (CSAR EQL #0) AND (INTREQ OR NOT RUNFLG) =>
[012]                 CSAR←4)          !interrupt/halt vector
[012]           NEXT  CSIR←CSTORE[CSAR]
[012]           NEXT  CSAR←(CSAR + 1)<11:0>
[012]           NEXT  MXEQ);
[012]   !
[012]   !in repeat loop
[012]   !
[012]   (DECODE CNTR<3:0> EQL "F =>
[012]           (CNTR←(CNTR + 1)<7:0> NEXT       !keep looping
[012]            CSIR←CSTORE[CSAR] NEXT
[012]            MXEQ);
[012]           (CSIR←CSTORE[CSAR]  NEXT         !last time thru
[012]            CSAR←(CSAR + 1)<11:0>  NEXT
[012]            REPEAT←0  NEXT                  !reset repeat flag
[012]            MXEQ)))  NEXT
[012] MLOOP)
[012]
```

Figure 1 Main Body of the HP21MX ISPL Description

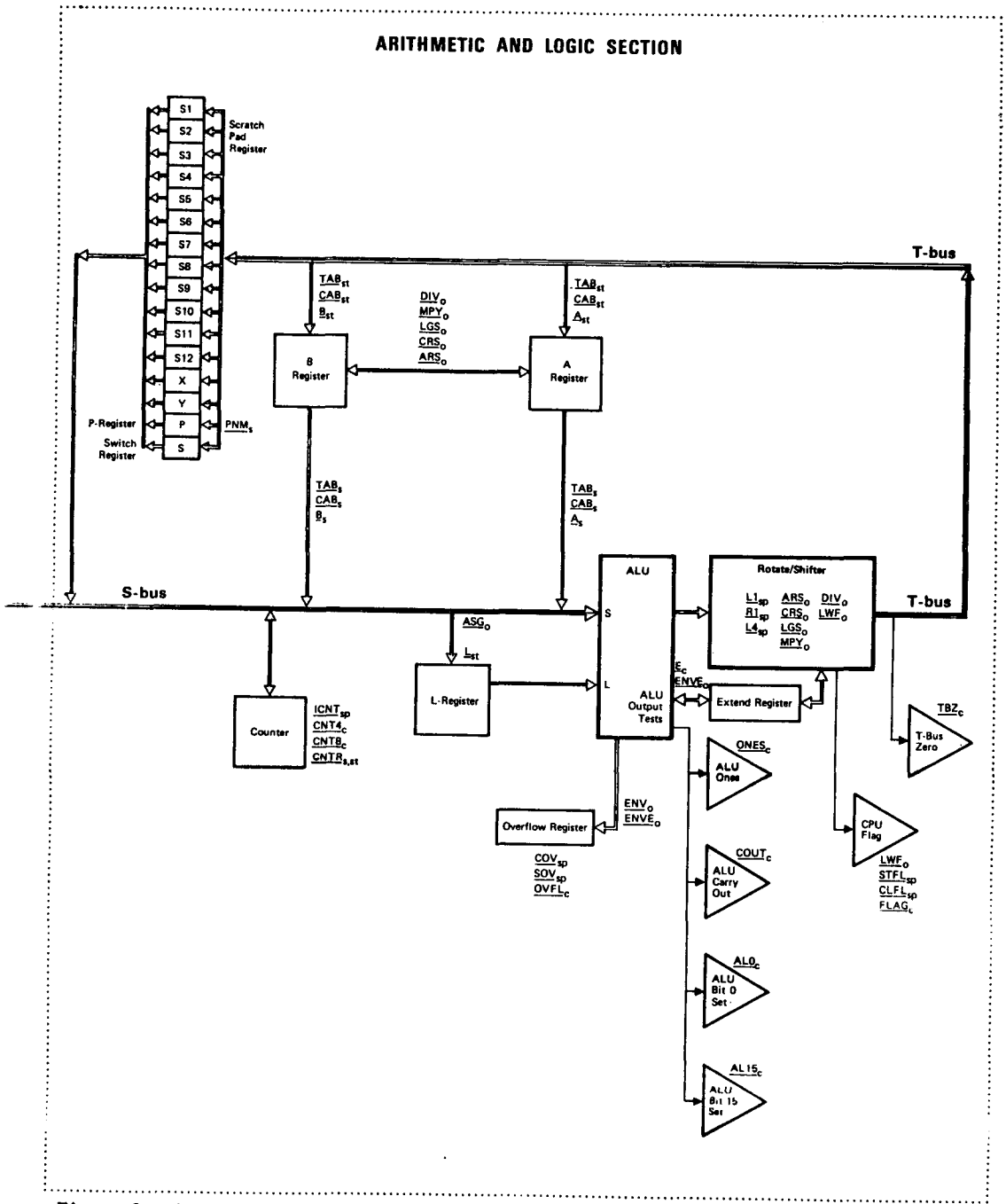

Figure 2 Block Diagram of the HP21MX processor(from [Hewlett-Packard 74])

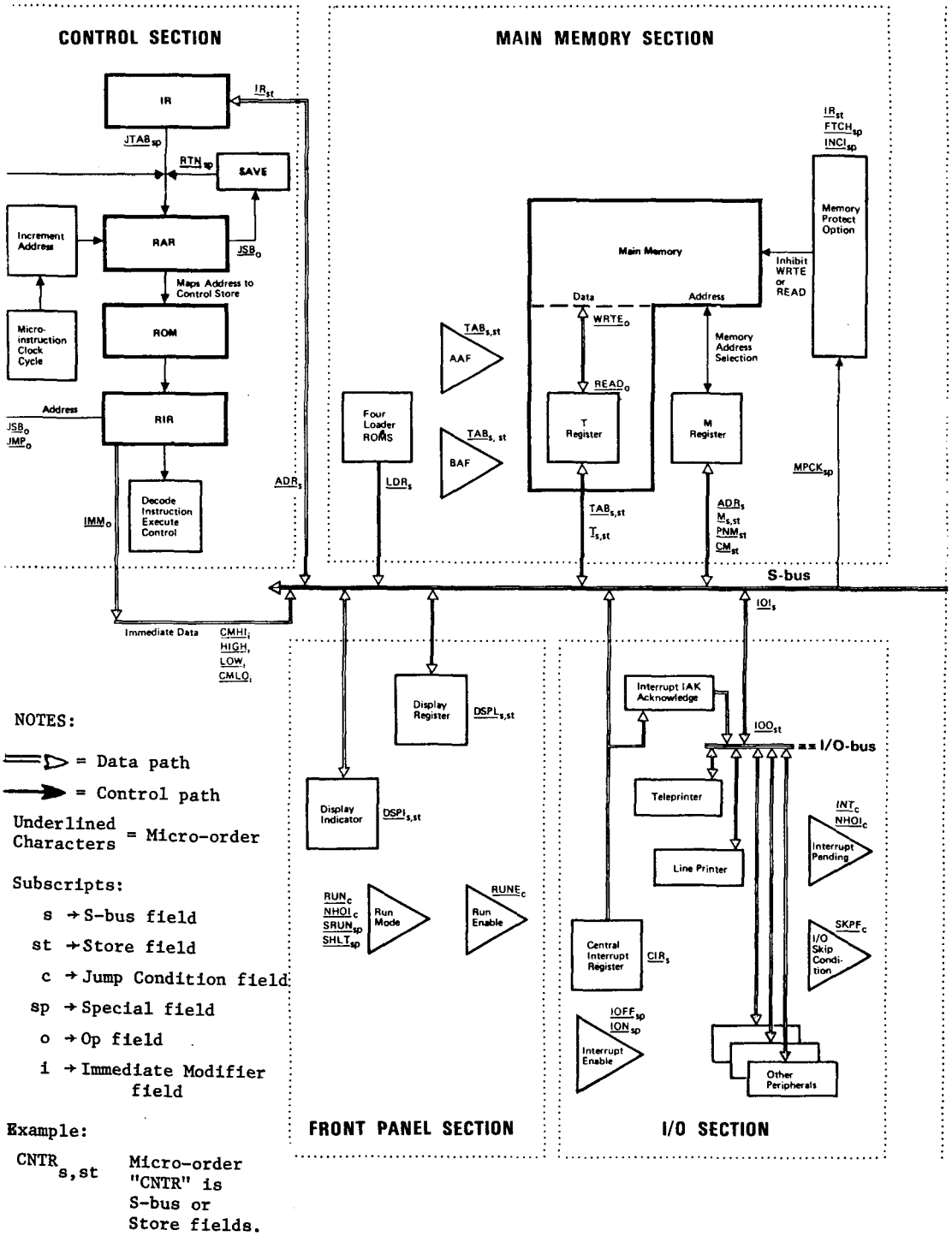

Figure 2 (Continued)

CLASS I - DATA TRANSFER / ALU OPERATIONS

23 20	19 15	14 10	9 5	4 0
OP	ALU	S BUS	STORE	SPECIAL

GENERAL OPS: NOP, LWF, WRTE, ASG, READ, ENV, ENVE
RESTRICTED OPS: ARS, CRS, LGS, MPY, DIV

ALU: 74181 Function code
S-BUS: Source register
STORE: Destination of T-BUS/S-BUS
SPECIAL: Misc. control signals

CLASS II - IMMEDIATE DATA

23 20	19 18	17 10	9 5	4 0
IMM	MOD	OPERAND	STORE	SPECIAL

MOD: Determines byte position and whether data is complemented

CLASS III - CONDITIONAL JUMP

23 20	19 15	14	13 5	4 0
JMP	CONDITION	R	OPERAND	CNDX

CONDITION: Selects tests
R: Reverse condition
OPERAND: Low 9 bits of address

CLASS IV - UNCONDITIONAL JUMP

23 20	19 17	16 5	4 0
OP	0 0 0	ADDRESS	MODIFIER

OP: JMP, JSB
MODIFIER: Specifies (conditional) modification to address

Figure 3 HP21MX Microinstruction Formats

```
>>SETVAL PREG=10000
>>SETVAL RUNFLG=1
>>SETVAL CSAR=0
>>110 LINES READ
>UTRACE ALL
>DTRACE FOO
>VALUE AREG

AREG    =#0
>VALUE BREG

BREG    =#1
>VALUE MEMORY[105:110]

MEMORY [#105]=#11
MEMORY [#106]=#7
MEMORY [#107]=#17
MEMORY [#110]=#0
>VALUE PREG

PREG    =#10000
>START MLOOP

@ MLOOP   +#2    REPEAT =#0
@ MLOOP   +#12   CSIR   =#4407471₂
@ MLOOP   +#14   CSAR   =#1
@ LDSBUS  +#111  SBUS   =#10000
    :

SIMULATION COMPLETED

RUN TIME(10 usec units)=992505
RTM OPS EXECUTED=3248

>VALUE AREG

AREG    =#31
>VALUE BREG

BREG    =#1
>VALUE MEMORY[105:110]

MEMORY [#105]=#11
MEMORY [#106]=#7
MEMORY [#107]=#17
MEMORY [#110]=#31
```

SETVAL sets values prior to simulation. It is a user command.

All values are being traced except FOO

The user asks the values of the AREG and BREG, which are 0_8 and 1_8

The user requests memory values

The user starts the simulation at MLOOP. At two statements beyond the MLOOP label, REPEAT is used

The simulation is over

The user inspects some locations

Figure 4 Output From the HP21MX Simulation