MIDAS Preliminary Description

Robert F. Gurwitz Read T. Fleming

Program in Computer Science

Brown University

Box F

Providence, R.I. 02912

TABLE OF CONTENTS

1 INTRODUCTION

The purpose of this paper is to describe MIDAS, a Microprocessor Interpreter and Display and Animation System, a system that provides the user with a real time simulation and graphical display of that simulation. The system is intended for instructional use, to accquaint the user with the workings of a typical computer system based around a production microprocessor, the Intel 8080. It consists of a discrete simulation of such a system, and a display program which gives the user a real time animation of the system's operation. MIDAS provides facilities for the user to control both the simulation and the display interactively.

What follows is a description of the features available to the user to control the system, a description of the simulation, animation techniques, and some auxiliary functions that enable the user to input data into the simulated system.

MIDAS is implemented on the Brown University Graphics System (BUGS). The reader is assumed to be familiar with the components of that system. MIDAS makes use of a standard low-level graphics support software package, the SIMALE Standard Graphics Package.[1] Parts of the system are written in ALGOLW and the rest in META4A Assembly Language.

---

[1] See Webber and Burns, The SIMALE Standard Graphics Package Preliminary Description, available through BUGS.

2 SYSTEM OVERVIEW

2.1 SIMULATION

The simulation part of MIDAS consists of a discrete simulation of a hypothetical system built around the Intel 8080 Microprocessor. This simulation maintains and updates a data base of system status. There is status information for each of the simulated devices of the system. In addition to the CPU, these include a Console, Memory, Clock, Keyboard Interface, Bus Control Unit (BCU), Status Decoder, Floppy Disk Controller, Direct Memory Access Controller (DMAC), and Interrupt Control Unit (ICU). These devices allow depiction of DMA and peripheral interface activity, as well as CPU-Memory operations. While the simulated CPU is based on an actual production device, all other devices correspond to no specific commercially available hardware, but have been designed to simulate the activity of typical hardware currently available.

The status information maintained by the system for each device is such that their operation can be simulated and displayed on a highly detailed level. This level extends to the state of all external registers and control lines of each device, but does not show the internal operation of the device. (For example, the internal bus transfers of the CPU are not simulated.) However, an attempt has been made to retain a high degree of accuracy in representing the relative timing of all operations. Also, the simulation is designed so that other devices than those described herein may be added to the system at some later date. While an attempt was made to keep the simulation program as general as possible, the prospective user should note that replacing any of the devices would require a relatively extensive amount of modification to the simulation program. However, the simulation and display programs do operate somewhat independently, so that with proper modification of the graphic data base used by the display, any simulation could be used.

Following is a brief description of each of the devices in the system. It includes an operational description and a list of what is displayed for that device.

2.1.1 THE CPU

As mentioned above, the CPU of the system is a simulation of the Intel 8080 Microprocessor. This is a processor that employs a word length of 8 bits and the capability for 16 bit addressing. A stack pointer is maintained for program subroutines. The CPU has a file of

6 user accessable registers that may be used in pairs for addressing operations. In addition, there is a Program Counter, Accumulator and a Temporary Register that is not directly accessable to the user, but which is used to hold intermediate results in some operations. There is also a complement of five comparison indicator flags (for sign, arithmetic carry, parity, zero and decimal overflow). The processor has 78 instructions, most of which are simulated (ultimately, all will be). All the above registers and flags are displayed, including a data register and address register (these terminate the address and data busses). The 10 control lines are also shown, as are the data and address busses.

2.1.2 MEMORY

The memory simulation consists of a 1K by 8 memory area maintained by the simulation. The memory is random access and assumed to have a cycle time faster than the CPU or any peripheral so that a "memory ready" control line is not used. An address register and data register are displayed, along with read and write control lines.

2.1.3 THE CONSOLE

A console is provided that is able to do DMA. Simulated on the console are a data and address registers, a switch register (whose input is a group of function keys available to the user), an Interrupt Instruction Register (necessary for the CPU interrupt scheme of the 8080) and an Interrupt Flag (set by the user via a function key). The console enables loading or displaying of any memory location, or any of the console registers.

2.1.4 CLOCK GENERATOR

The clock generator provides two clock pulses needed by the CPU. These are displayed with the two clock input lines to the CPU. There is also a timing display of the clock output.

2.1.5 BUS CONTROL UNIT AND STATUS DECODER

These two devices interface the CPU with the rest of the system. The BCU controls direct memory access (DMA) operations by devices external to the CPU. These currently

include the console and the disk DMA interface. Devices signal requests to the BCU which then determines, through a priority network, which one gets service. The HOLD and HLDA (Hold Acknowledge) lines of the CPU are used to stop it during the DMA transfer, and an "enable" signal is sent to the device to be serviced.

The Status Decoder receives the output of the CPU during T1 of the CPU's machine cycle. The output of this device determines what type of operation the CPU is performing. Through this, certain control signals needed by other devices (such as memory) are derived. A status register displays the status byte sent out by the CPU.

2.1.6 INTERRUPT CONTROL UNIT

This device takes input from each device that can interrupt the CPU, and through a priority network decides which device can interrupt the CPU. The ICU initiates the interrupt and sends an enable signal back to the device when the interrupt is acknowledged. The ICU's operation is displayed through its control lines.

2.1.7 DISK CONTROLLER AND DMA CONTROLLER

These two devices act as control for a floppy disk peripheral. The Disk Controller simulates a read or write to a disk, which corresponds to a 4K area of storage maintained by the simulation. Latency time is simulated by a fixed delay of several system cycles. A sector/track address register and seek flag are maintained by the controller, as well as a data register.

Interfacing the Disk to the rest of the system is a DMA controller. This is simulated with a data and address register, a count register and several control lines to the disk and to memory.

2.1.8 KEYBOARD INTERFACE

The Keyboard Interface is a device which is used to illustrate interrupt-driven I/O. Each time a key is struck, the character is sent to the keyboard interface and an interrupt is signalled to the CPU. An I/O interrupt routine can then read the character via an 8080 input instruction. The interface has a character register where incoming or outgoing data is displayed and interrupt request and acknowledge lines.

2.2 THE DISPLAY

This section describes the display used by MIDAS and the animation techniques used in the system. The display program operates on a static graphic data structure maintained by the SIMALE Standard Graphics Package. It also contains a structure that enables the basic elements of the display, the registers and flags, control lines and busses, to be altered dynamically. The display program responds to a series of commands generated by the simulation program. Each time the simulation program is invoked (referred to as a cycle), the simulation data structure is updated and a set of display commands is made up. These commands are interpreted by the display program and cause updating of the display data structure to take place. In addition, some emphasizing activity is also done, depending on the instruction, which causes the display to change and draw the viewer's eye to the change. These techniques are described below.

The use of such an interpretive program allows some degree of independence of the display program from the simulation. Thus, altering the display consists of altering the static display data structure.

2.2.1 THE BASIC DISPLAY FRAME

A standard schematic view of the system is the basis of the display. Contained within this schematic are several elements. For each device, a box appears on the display along with legends identifying the device and the lines eminating from it. In addition, the registers or flags of the device are displayed with an identifying legend and a representation of the data they contain. This representation is in the form of hexadecimal digits for registers, and an on/off indicator for flags. Tying the devices together are three types of data paths: single bit control lines, brightened when active; an 8 bit data bus, the two hexadecimal digits of data travelling across the bus when active; and a 16 bit address bus in which four hex digits of data travel when active.

This basic display frame is contained within a viewport. Outside this viewport is an area for status and command prompts generated by the system and the user. The user may choose to "zoom" in on any part of the basic display frame, on several levels of detail. This isaccomplished via a joystick which positions a window into which the basic display frame is projected. The user can also physically zoom in on any area of the display via a dial. As he moves in, more complex levels of detail appear (such as register data, flags, etc.). This frame is static. Within it, several techniques are employed in animating the operation of the system.

2.2.2 ANIMATION TECHNIQUES

As described above, there are several basic operations performed by the display program in displaying the data. These are as follows. When a control line changes status, this is indicated by a brightening of the affected line. When a register or flag changes status, this is indicated by a flashing of the affected data in the appropriate register or flag. The display program can also change data without flashing. This is useful when the system is initialized or a saved data file is read by the system (see 2.3.2). Finally, when data is transferred across a bus, the actual digits of the data are shown travelling along the bus. Using these four basic operations, along with the status display described in the next section, animation of the operation of the system is achieved.

2.2.3 STATUS DISPLAY

In addition to the basic display frame, several prompt areas are used to inform the user of the current state of the system. These status messages include a description of the current machine cycle of the CPU and a description of the cycle type the CPU is in. A clock display gives the user an indication of the relative speed of the system.

A display of program status is also given. This includes what mode the system is in (see 2.3.1), the current data file being executed (see 2.3.2), and the current command line the user has typed in.

2.3 THE CONTROLLER

In addition to the simulation and display parts of the system, MIDAS also has a controller which performs utility operations important in the use of the system. These functions include initialization of the system, a command processor for a small command language which is the user's interface with the controller, a small loader program which enables the simulated memory area to be loaded with programs and data for execution by the system, and real time operation of the simulation, which allows for operations which would be unfeasable in normal operation. The controller also performs disk I/O, needed for saving and restoring system status at any time.

2.3.1 MODES OF OPERATION

There are three operating modes which MIDAS can be in during execution. These are Command Mode, Real Time Mode, and Normal Mode. Each of these modes allows for utilization of different parts of the MIDAS system. In Command Mode, where the system is placed on invocation of the system or any time the display is stopped, commands may be entered through the alphanumeric keyboard. These commands can invoke the utility functions of the system or restart the system in any of the other two modes. In Command Mode, the simulation is stopped and the display remains static at the point where Command Mode was entered. The commands accepted by the system are described below.

In Normal Mode, the simulation and display programs operate in such a way that the simulation is called once every cycle and the display program then executes for a period of time determined by the user via a dial. Thus, the speed at which the system operates is at a level where the user can follow the display at his own pace. Normal Mode is entered from Command Mode.

Real Time Mode is also entered from Command Mode. In this mode, the simulation proceeds at a rate closer to that of the operation of the simulated system in real time. The display is disabled. This mode is meant for letting the simulation operate in a "useful" way (such as reading or writing a block of data from disk, or executing any program in simulated memory in real time).

Mode transition and program control is accomplished with the function keys and the alphanumeric keyboard. Both device serve in dual roles. During Normal and Real Time Modes, these devices are part of the simulation as peripherals in the system. In Command Mode, these devices communicate with the controller.

2.3.2 THE COMMAND LANGUAGE

The MIDAS command language gives the user a convenient way of interactively communicating with the system via the alphanumeric keyboard and displaying and altering the system status. Each one line command consists of a function keyword followed by operands, followed by a carriage return.

For example, "SET CR HL 03E8" tells the controller to SET CPU Register pair HL to the hexadecimal value 03E8. Similarly, the SET command can be used to alter flags, non-CPU registers, and memory locations. The DISP command displays these entities without changing them. SAVE and RECALL are used to save and recall status information of the entire system in a disk file, allowing a run to be

saved between invocations of MIDAS. Since up to five unique files can be kept in this manner, SAVE/RECALL can be used for "instant replay". RESET will reset a given part of the system (e.g., deselect an I/O device), and CLEAR will reinitialize everything by clearing memory and resetting all system flags and registers. This is the state the system is in when it is invoked. Initialization is accomplished from a disk file. The simulation status information is read in as is the graphic data structure. The graphic data structure is kept on disk and is initialized with the Standard Graphics Package Subroutines. There is also a DEBUG command which is meant to aid those maintaining MIDAS.

2.3.3 THE LOADER

Programs and blocks of data ca be loaded from disk using the LOAD command. This relieves the user of the task of having to load large amounts of information a few bytes at a time via the SET command. LOAD takes as its operands the name of the file to be loaded and the memory location to start loading it at. FILE will save a specified section of memory under a given name. ERASE and INFO are used to maintain files. Note that no "relocation" is performed during loading; i.e., a program should be LOADed at the same address it was SAVEd from.