Document Nbr: CSI426/kayproII-RS2.06
Date: December 5, 1998
Copy Nbr: ____

REQUIREMENTS SPECIFICATION DOCUMENT

FOR THE

CSI426/KAYPRO II EMULATOR

Distribution List

Name	**Organization**	**Signature**
Dr. Charles Howerton	**Metropolitan State College**	______________________________

Table of Contents

Tables and Figures

Figure 1, Kaypro II Overview 6
Figure 2 Kaypro II Emulator Java Applet Transfer 8
Figure 3 Kaypro II functional View 12
Figure 4, Port Emulation.......... 13
Figure 5 Kaypro II Bank Switching 15
Figure 6, Memory Mapped Video 16
Figure 7, Logical Vs. Physical Screen Mapping 17
Figure 8, User Interface with Debug Off 18
Figure 9, User Interface with Debug On.......... 20
Figure 10, View Options.......... 22
Figure 11, SIO Indexing Scheme 24

1. Introduction

This section contains the overview, system identification, and scope of the Requirements Document.

1.1 Overview

This document describes the software specification for the Kaypro II emulator.

This document presents an overview of the Kaypro II emulator. The emulator is a Java based implementation of a Kaypro II computer system. Java is a language that allows remote programs to be executed on a client computer via the World Wide Web. The Kaypro II emulator resides on a remote host machine. The user may run the emulation on a compatible browser from their own computer system.

The implementation features a number of convenient and useful features, including:

- Printer port simulation
- Debugging:
    - Hardware style breakpoints
    - Opcode level debugging
    - CPU register display
- Memory dump utilities
- Dual virtual diskette drives with CP/M pre-loaded
- Real and fast mode video options

In addition to the above, the system shall be coded in such a way that it is expandable. This expandability requires changes to the source code. The modular nature of the implementation allows easy modification.

1.2 Scope

The Kaypro II emulator utility shall be a new development effort.

2. System Requirements

2.1 System Definition

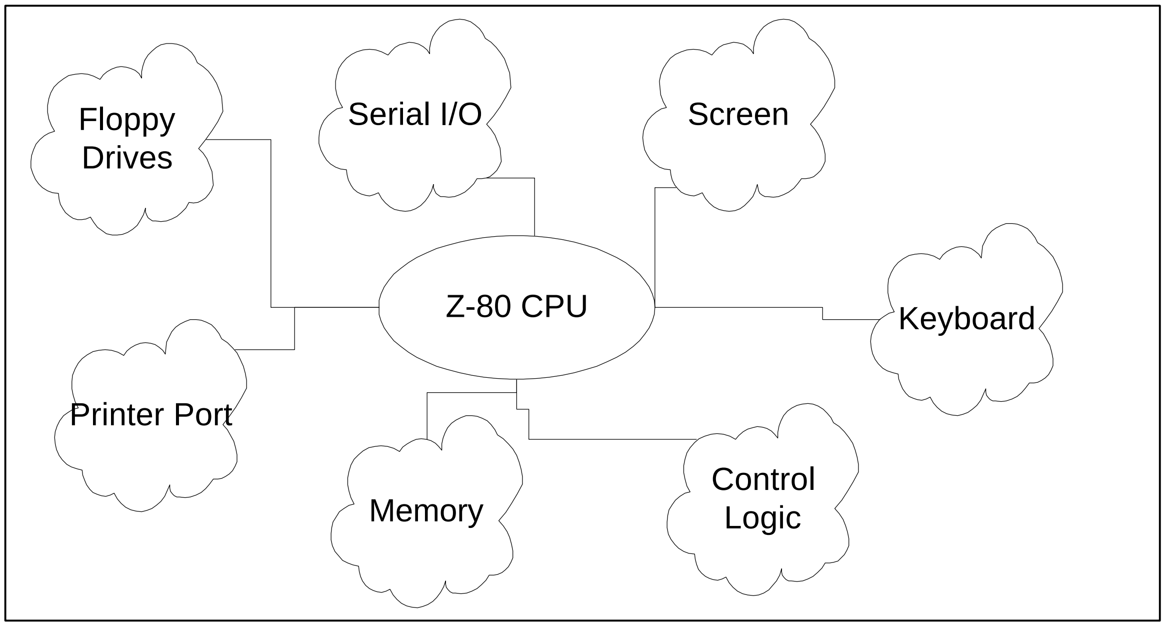


Figure 1, Kaypro II Overview

2.1.1 Concept of Operations

The Kaypro II is a Z-80 based computer system. It contains dual floppy drives, serial I/O, a monochrome screen, keyboard, a printer port, memory and control logic.

The Z-80 processor executes pre-programmed instructions read from memory. Serial I/O provides an interface to serial devices. Serial I/O also provides interface to the built-in keyboard and system speaker.

The Kaypro II contains a monochrome screen. The screen has no real graphic capability, but can display inverse characters.

The Kaypro II includes two built-in floppy disk drives.

The printer port enables communication with a Centronics compatible printer. A secondary printer port acts as an interface for controlling internal functions such as:

- Floppy disk selection
- Floppy drive motor control
- RAM/ROM/video RAM bank selection
- Printer control signals

The Kaypro II supported 64K of RAM. In addition, the system also included a system ROM and memory mapped video.

The Kaypro II control logic included a dual baud-rate generator for serial data rate adjustments, a character generator ROM, and other discrete control logic.

2.1.2 Assumptions and Constraints

The Kaypro II emulation project focuses on the internal function of the Kaypro II hardware. The goal of the project is to provide a working emulation of an actual Kaypro II computer system. The emulator will be capable of running most Kaypro II software.

- It is assumed that the user of the system is familiar with the CPM operating system.
- It is assumed the user is familiar with the operation of a Kaypro II computer.
- It is assumed that the user is familiar with common debugging techniques.
- It is assumed that the user is familiar with Windows 95, NT 4.0, or similar GUI's.
- It is assumed that the Java browser is JDK 1.1.6 or above

2.2 Functional Requirements

2.2.1 Function

- The emulation program shall emulate the Kaypro II model of the Kaypro product line
- The user shall manipulate the emulated version just as they would the original
- The emulation shall contain debugging features, in addition to the original functionality

2.2.2 Expandability

The system shall be expandable. That is, it's design shall be easily upgraded. This allows improvements, variations and upgrades to be easily coded and implemented.

- The Kaypro II emulation shall be coded in such a way as to facilitate easy additions and expansions to the system.
- The emulation shall be coded in a modular way; such as logical Java classes

2.2.3 Platform

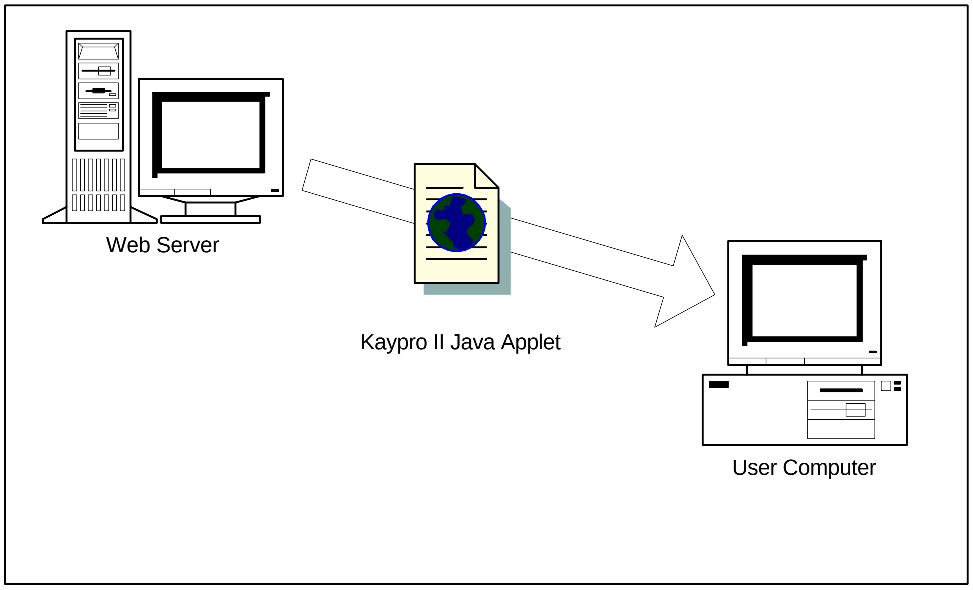


Figure 2 Kaypro II Emulator Java Applet Transfer

The Kaypro II emulation shall be implemented as a Java applet. Java applets reside on remote servers (see Figure 2). When a user browses an HTML web page containing reference to the emulator applet, the applet byte code is transferred to the User's computer and executed[1].

- The Kaypro II emulation shall be implemented in Java
- The implementation shall be pure Java (e.g. No Microsoft extensions).
- The Kaypro II emulation shall be implemented as an applet.
- The Kaypro II emulation shall be made available via the world wide web
- The Java interface shall be kept minimal, to afford quicker load times
- The Implementation shall use JDK 1.1.6

[1] See Sun Micro's description of the Java environment and language.

2.2.4 Z-80 CPU Emulation

- The Kaypro II utilizes the Z-80 Processor. The CPU emulated shall be the Z-80
- The CPU instruction set shall conform to those presented in the *Zilog Z80 Microprocessor Family User's Manual, Part number Q1/95 DC 8309-1.*
- The RLA command as documented in the Zilog user's manual contains an error in the rotate diagram. The emulation shall support shift left, whereas the manual depicts shift right.

2.2.4.1 Z-80 Features

The Z-80 CPU contains a number of notable features. These include:

- One 8-bit Accumulator
- One 8-bit flags register
- Six 8-bit general purpose register, that can be mapped to three 16-bit registers
- An alternate register set
- An interrupt vector register
- Two 16-bit index registers
- One stack pointer
- One program counter

2.2.4.2 Z-80 Instruction Set

The Z-80 instructions are a superset of the 8080.
A summary of the Z-80 instruction set is included in appendix A.

The Z-80 instruction set consists of the following groups of operations:

- Load and exchange
- Block transfer and search
- Arithmetic and Logical
- Rotate and shift
- Bit Manipulation
- Jump, Call and return
- Input and Output
- CPU Control (NOP, HALT, etc)

2.2.4.3 Z-80 Arithmetic Logic Unit (ALU)

The Z-80 ALU supplies the following functions:

- Add
- Subtract
- Logical AND
- Logical OR
- Logical Exclusive OR
- Compare
- Shift and rotate
- Increment and decrement
- Bit operations

- The ALU shall be implemented as a function of the CPU. It may not be implemented separately.

2.2.4.4 Z-80 Addressing modes

The Z-80 CPU shall support the following addressing modes:

- Immediate, where data is explicitly specified within the instruction (8 bit)
- Immediate extended, where data is specified within the instruction (16 bit)
- Zero page, where a single byte instruction may call one of eight zero-page locations
- Relative, where the following byte specifies a relative address
- Extended, where a 16 bit value specifies the location of an indirect address
- Indexed, where an index register and an offset specify an absolute address
- Register addressing, where a particular register specifies an address location
- Implied addressing, where the opcode automatically implies a CPU register
- Register indirect addressing, where the register contains an indirect address reference
- Bit addressing, where memory or registers may directly manipulate individual bits

2.2.4.5 Main Registers

The Z-80 emulation shall include the following main registers. These registers shall be contained within the processor, and be accessible to the CPU instructions. The alternate instructions are accessible via a Z-80 swap command. This command swaps the main and alternate register sets. Unless swapped, the alternate register set is not accessible to the Z-80 instructions.

Main Register Set		**Alternate Register Set**	
Accumulator A	Flags F	Accumulator A’	Flags F’
B	C	B’	C’
D	E	D’	E’
H	L	H’	L’

Table 1, Z-80 Primary Register Set

2.2.4.6 Special Purpose Registers

Z-80 special purpose registers shall be included in the emulation. These registers assist in indirect addressing via Z-80 instructions, specifically, the IX and IY index registers. The program counter controls program execution, while the stack pointer register controls stack operations. The Z-80 CPU includes instructions for stack manipulation.

Special Purpose Registers	
Interrupt Vector I	Memory Refresh Register R
Index Register IX	
Index Register IY	
Stack Pointer SP	
Program Counter PC	

Table 2, Z-80 Special Purpose Register Set

- The Interrupt vector, I, is used for mode 2 interrupts (described below). It shall be implemented.
- The Memory refresh register is used for dynamic memory refresh. It is incremented each time an instruction is executed. Dynamic memory is a function of hardware implementation, and is not needed. The memory refresh register, R, shall not be implemented.

2.2.4.7 Non-maskable Interrupts

The Z-80 supports one non-maskable interrupt. This interrupt is executed when the NMI line of the CPU is activated. The NMI interrupt forces a CPU restart (call) to location 0x66 upon the completion of the current instruction.

- The non-maskable interrupt may not be disabled.

2.2.4.8 Maskable Interrupts

The Z-80 CPU supports 3 modes of maskable interrupts: mode 0, mode 1, and mode2

- Maskable interrupts may be disabled via CPU instruction.

2.2.4.8.1 Mode 0

An interrupt is executed when the INT line of the CPU is activated. The INT mode 0 interrupt forces execution of the instruction placed on the bus by the interrupting device. The execution of the device-supplied instruction takes place upon the completion of the current instruction.

- Mode 0 shall be implemented.

2.2.4.8.2 Mode 1

An interrupt is executed when the INT line of the CPU is activated. The INT mode 1 interrupt forces a CPU restart (call) to location 0x38 upon the completion of the current instruction.

- Mode 1 shall be implemented.

2.2.4.8.3 Mode 2

An interrupt is executed when the INT line of the CPU is activated. The INT mode 2 requires that the programmer setup a table of 16 bit service routine addresses. When an interrupt is generated, a 16 bit address is created. This address points to an element in the table.

The upper 8-bits of the address is specified by the programmer, and stored in the I register. The lower 8-bits are supplied by the interrupting device. This address is used by the CPU to index into the programmer-supplied table. The index points to the address of the interrupt service routine.

The execution of the interrupt service routine takes place upon the completion of the current instruction.

- Mode 2 shall be implemented.

2.2.4.9 Reset

The Z-80 CPU shall support implementation of the RESET line. When this line is activated, the CPU shall force a jump to location 0x00 upon completion of the current command.

2.2.4.10 Bus Timing and Signals

Bus timing and associated signals shall not be implemented. Only the function of the CPU shall be implemented. Exact CPU speed shall not be governed, except within the limits of the executing hardware and software.

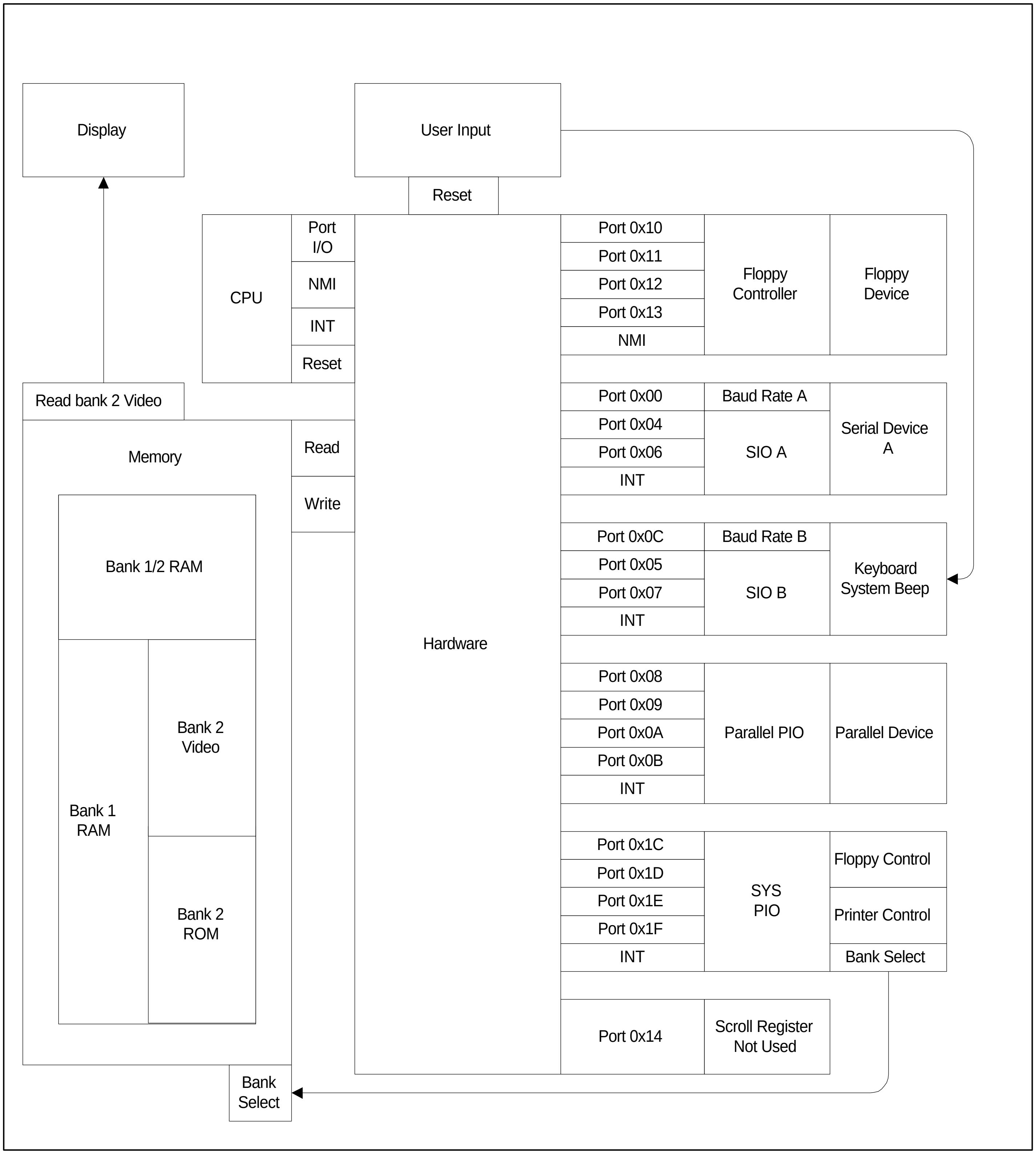


Figure 3 Kaypro II functional View

2.2.5 Hardware Emulation

Figure 3, Kaypro II functional view, shows the Kaypro II architecture. Central to the architecture is the hardware. The hardware represents the communication mechanism between the various system components. The hardware represents discrete chips and connection logic within the actual Kaypro II. The hardware shall be a link from the CPU to all the other devices. The hardware connects the following devices.

- Keyboard
- Screen
- Memory (Ram/Video/RAM)
- Bank switching
- I/O Ports
- Disk Drives
- Motor control, indicator lights, etc

The hardware passes the following signals (messages):

- User action
- NMI
- INT
- Reset
- Read memory
- Write memory
- Port commands

2.2.6 Port Emulation

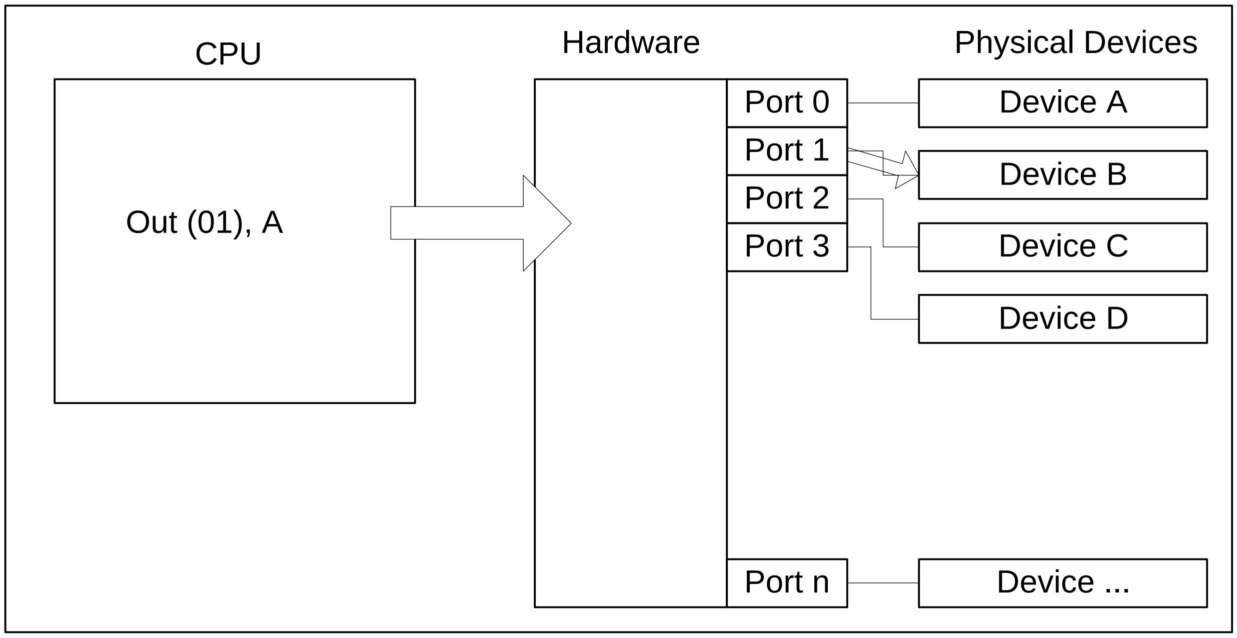


Figure 4, Port Emulation

An important feature of the bus is the implementation of ports. Ports allow the Z-80 to communicate programmatically with external hardware devices.

Figure 4, port emulation, illustrates the logical port operation of a Z-80 OUT command. In Figure 4, a Z-80 OUT command is sending data contained within the accumulator (A) register to a physical device labeled B. Note that device B is physically connected to port 1. The device would receive the value contained within the accumulator. The Z-80 IN command can retrieve data from a physical device in a similar manner.

IN and OUT commands allow data to be transferred to external devices. Examples of Kaypro II external devices include:

- Floppy disk controller/Drive
- Serial I/O
- Keyboard
- Beeper
- Parallel port/printer
- Baud rate generators

The Z-80 supports numerous ports (up to 256 ports are available). The hardware designer maps ports to physical devices. The following ports are utilized within the Kaypro II:

Usage	Kaypro Name	Port
Baud rate Generator A, baud rate for Kaypro external serial port	Baud rate port	0x00
Serial port A, Kaypro external serial port	Bit port	0x04
Serial Port B, Keyboard	Bit port	0x05
Serial port A , Control register	Bit port	0x06
Serial Port B, Control register	Bit port	0x07
PIO A, Data register (write to printer, used to send data to printer)	Bit port	0x08
PIO A, Control register (write to PIO, used to program PIO)	Bit port	0x09
System PIO, Data (not used)	Bit port	0x0A
System PIO, Control (not used)	Bit port	0x0B
Baud Rate Generator B, baud rate for keyboard	Baud rate port	0x0C
Floppy controller, Status/Command register	Floppy port	0x10
Floppy controller, Track register	Floppy port	0x11
Floppy controller, Sector register	Floppy port	0x12
Floppy controller, Data register	Floppy port	0x13
PIO A, Data register	Bit port	0x1C
PIO A, PIO control register	Bit port	0x1D
System PIO, Data register (not used)	Bit port	0x1E
System PIO, PIO control register (not used)	Bit port	0x1F

Table 3, Kaypro II Port Usage

2.2.7 Memory Emulation

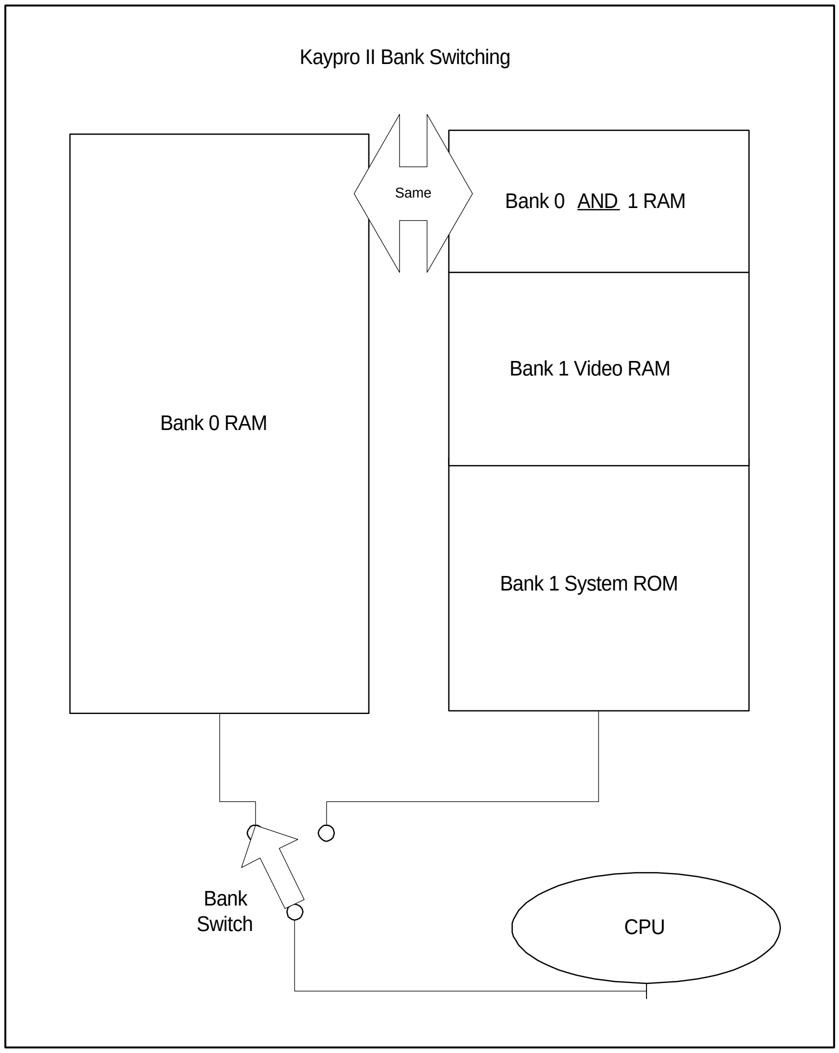


Figure 5 Kaypro II Bank Switching

The Kaypro II utilizes two banks of memory. Bank 0 contains 64K of linear RAM. Bank 1 contains the video and system ROM. Notice from Figure 5 that bank 0 and bank 1 share high memory.

One way to understand the Kaypro II banking scheme is to visualize a switch (see Figure 5). The CPU executes instructions from memory. The memory that the CPU sees is determined by the bank switch. Depending on the position of the bank switch, the CPU will operate on data from either bank 0 or bank 1. The bank switch is thrown electronically. This allows the Kaypro II to support 64K programs, while still supporting memory mapped video, and bootstrap ROM.

Note that the upper portion of bank 0 and bank 1 share bank 0's RAM. This allows a single program to operate in both memory domains.

Bank 1 contains ROM as well as video. The ROM, referred to as the "System ROM," is contained within a 2716 EPROM. This EPROM is pre-programmed with utility routines, as well as a bootstrap loader. When the system is reset, bank 1 is selected, and code from within the System ROM is executed. This code loads the operating system from floppy and initializes the system. The complete memory map for the Kaypro II is shown in Table 4.

Bank	Type	Range
0	System RAM	0x0000 - 0xFFFF
1	System ROM (2716)	0x0000 – 0x2FFF
1	Video RAM	0x3000 – 0x3FFF
1	System RAM[2]	0x4000 – 0xFFFF

Table 4, Kaypro II Memory Map

- The emulated RAM shall be 64K bytes.
- The emulated ROM shall be 4K Bytes (although 12K of address space is reserved)
- The emulated Video RAM shall be 4K Bytes.
- System ROM shall be extracted from the original Kaypro II 2716 EPROM, converted to programmatic form, and inserted into the emulator code.

*Note: Video and System ROM actually occupy only 4K. There is a void above each of these areas. This area can be occupied by larger ROM, for example. The Kaypro II has a jumper that allows upgrading the base unit to a 2732 ROM. It has been reported that some Kaypro systems mirror System ROM (duplicate electronically). That is, System ROM repeats within the void space.

2.2.7.1 Video and Memory

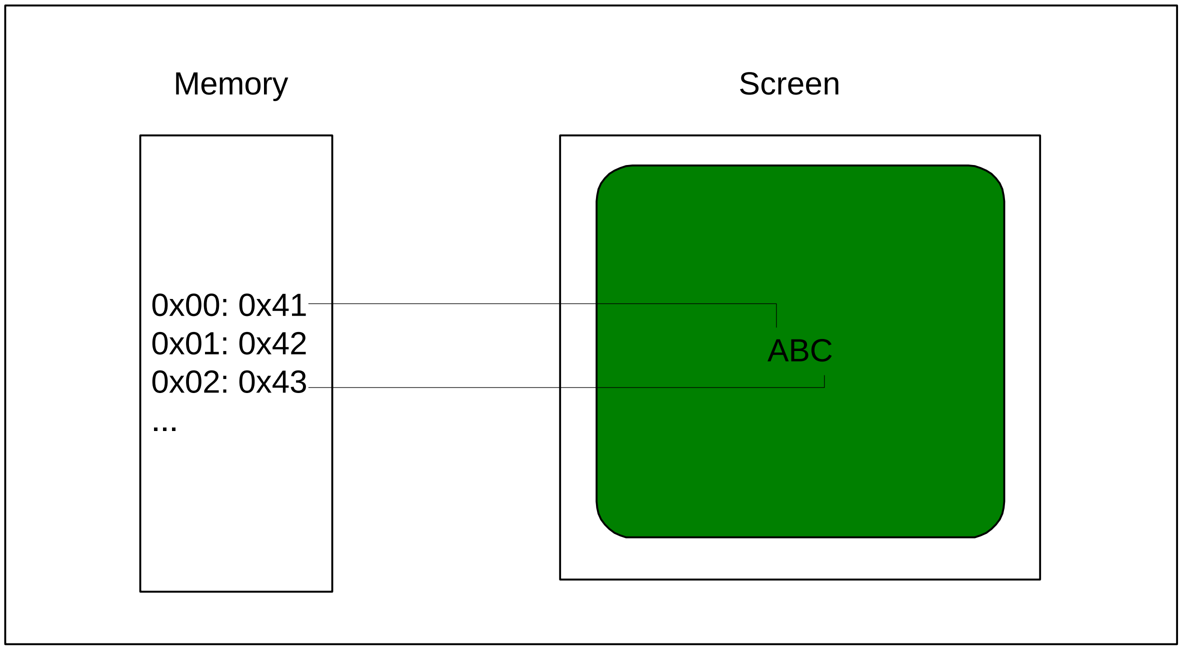


Figure 6, Memory Mapped Video

[2] As noted in the text, bank 1 system RAM is physically the same as bank 0 system RAM. They are logically and electronically the same.

Bank 1 contains "memory mapped video." Memory mapped video associates one byte in memory to one byte on the video screen. To display a character to the Kaypro II screen, the CPU switches to bank 1, and writes to the memory mapped video RAM. Each byte in video RAM associates to a character position on the video screen.

For example, location 0 in video RAM may associate to screen character location 0,0. Location 1 in video RAM may associate to screen character location 1,0 etc.

The physical screen is automatically updated by the Kaypro II hardware at regular intervals. Values stored in the memory-mapped video RAM are displayed as ASCII values on the Kaypro II screen. Figure 6 is an example of memory mapped video.

- The emulated video RAM shall be 4K bytes
- The emulator shall support an 80 character per line by 24 line text screen
- The emulator shall support memory-mapped video RAM banking

2.2.7.1.1 Logical and Physical Screen Mapping

As mentioned above, the screen video is memory mapped. That is, the video RAM has a one-to-one relation to the video screen. The mapping is somewhat different that would be expected (see Figure 7).

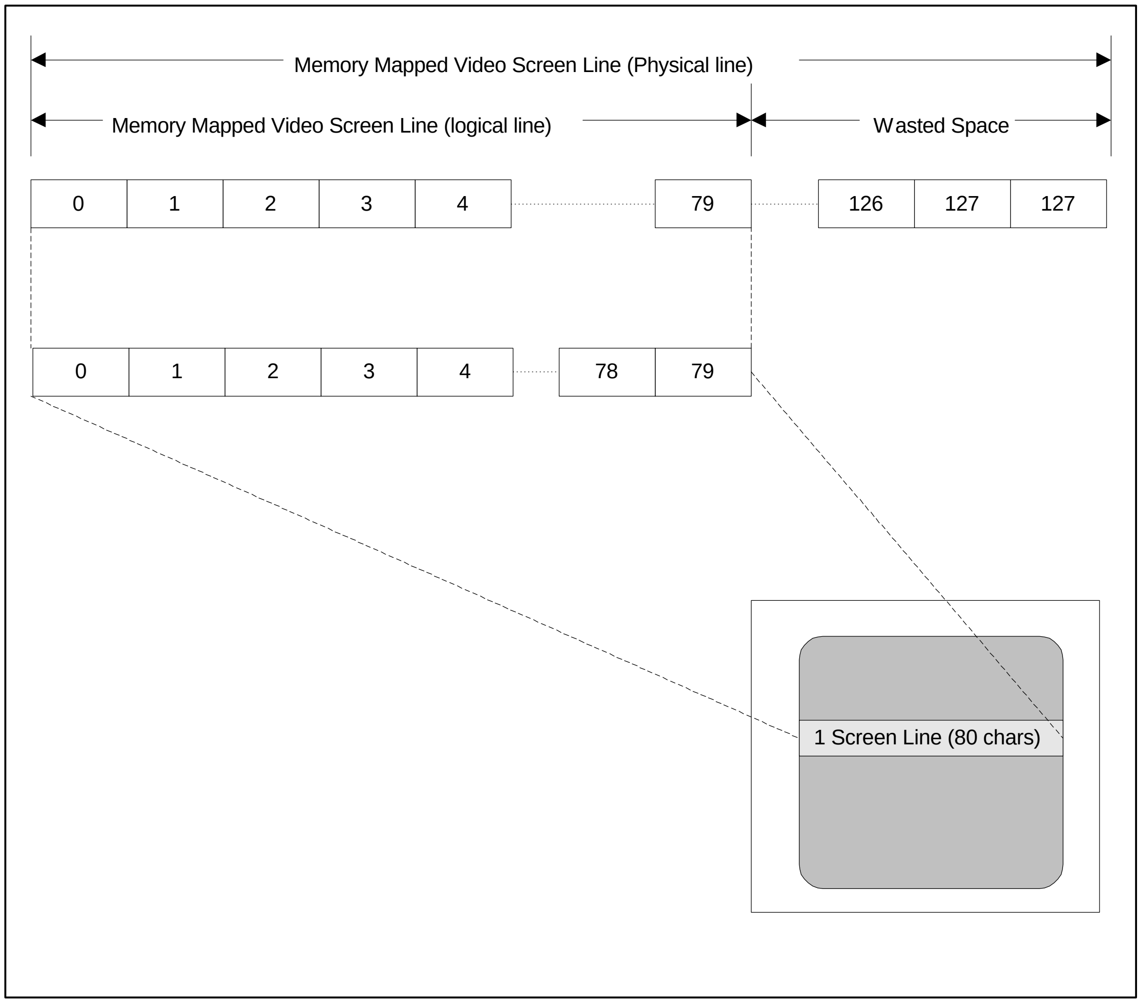

Memory Mapped Video Screen Line (Physical line)
Memory Mapped Video Screen Line (logical line)
Wasted Space
0
1
2
3
4
79
126
127
127
0
1
2
3
4
78
79
1 Screen Line (80 chars)

Figure 7, Logical Vs. Physical Screen Mapping

The Kaypro II video RAM area devotes 128 Bytes for each horizontal screen line (see Figure 7). Only the first 80 bytes are actually displayed, with the rest being unused. One reason for the wasted space may be the nature of the Kaypro II's discrete character generation circuitry. The extra 48 bytes may have been used to compensate for horizontal retrace.

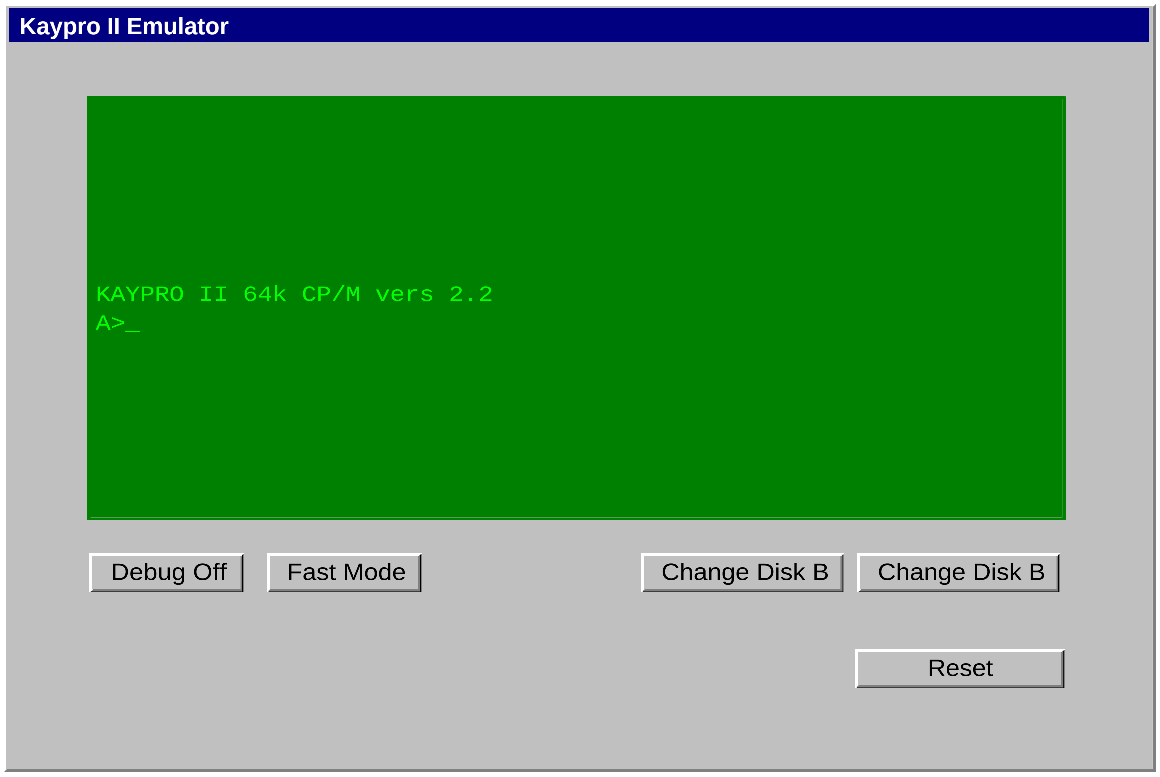
- The Kaypro II emulation shall mimic the logical and physical screen mappings of the Kaypro II.

2.2.7.1.2 Fast and Real Mode Display

The characters displayed on the Kaypro II screen are generated by a 2716 character generator EPROM. This ROM defines the dot grid for each character displayed. The Kaypro II emulator shall provide two methods of display: fast mode, and real mode.

- Fast mode will use a typical Java text box to display characters. Fast mode will not attempt to emulate the exact Kaypro II character generator ROM.
- Real mode will display characters as they are defined in the Kaypro II character generator ROM. Real mode will require extracting data from the character generator ROM, converting it to programmatic form, and inserting it in the emulator.

2.2.8 User Interface



```
KAYPRO II 64k CP/M vers 2.2
A>_
```

Figure 8, User Interface with Debug Off

The interface for the Kaypro II emulation shall allow the user to operate the Kaypro II emulation in a manner that closely resembles that of the original Kaypro II.

The user shall have the ability to select debugging options and control functions of the Kaypro II computer (i.e. changing disks, resetting). The interface shall have a screen that allows the user to input data and receive data from the Kaypro II emulation.

- The final screen viewing shall be acceptable on a 640 x 480 screen. The goal shall be to fit the entire display on such a display.

The basic interface is shown in Figure 8. This interface shall be displayed when the emulation is started. It shall contain a number of buttons that perform the following functions:

- Toggle Debug Mode On/Debug Mode Off
- Toggle Fast Mode/Real Mode
- Reset the emulation
- Change the virtual disk A
- Change the virtual disk B

2.2.8.1 Debug On/Off

The Debug Mode On/Debug Mode Off allows the user to toggle between debug mode, and normal system operation. When debug mode is on, additional buttons will appear (Figure 9). These buttons allow the user to manipulate the emulation in a low-level debug mode.

- The Debug Mode Off is default when the emulator is started.

2.2.8.2 Fast Mode/ Real Mode

The Fast Mode/Real Mode button toggles between fast output mode or real output mode. In fast mode the emulator shall output text using an available font to a text area. In real mode the output of the emulator shall be generated graphically using scan lines and a character ROM.

- The fast mode is default when the emulator is started.

2.2.8.3 Reset button

The Reset emulation button resets the emulation. This button has the same function as the button on the back of the Kaypro II computer. The reset button activates the RESET line of the Z-80 CPU.

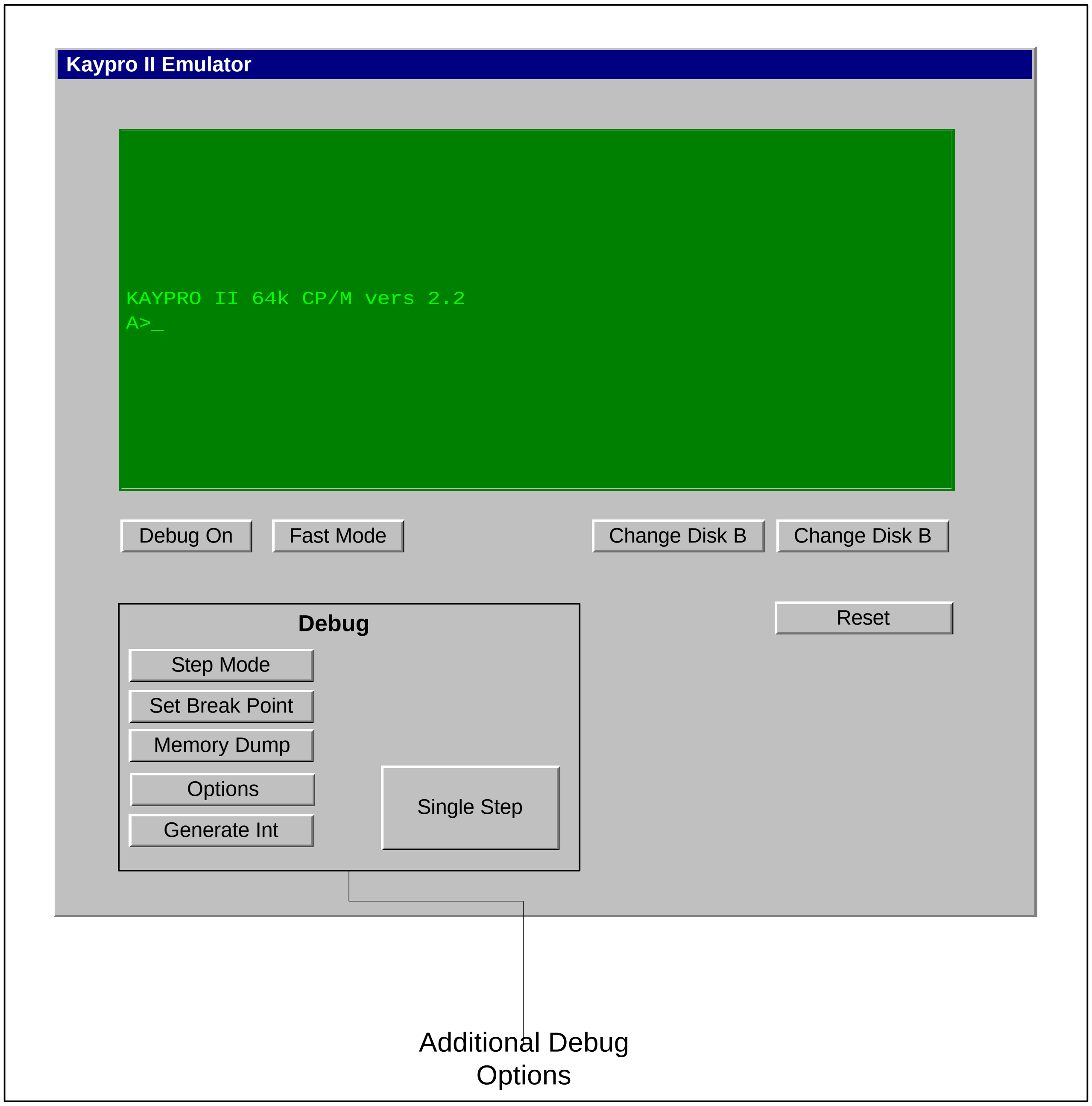
2.2.8.4 Change Disk A and B

The Change Disk A button shall prompt the user for the location of the virtual disk image file that shall be used in virtual disk drive A.

The Change Disk B button shall prompt the user for the location of the virtual disk image file that shall be used in virtual disk drive B.

2.2.8.5 Storage

The Kaypro II emulator shall store virtual diskettes to a specified server.



```
KAYPRO II 64k CP/M vers 2.2
A>_
```

Figure 9, User Interface with Debug On

The user may also select a debug mode. This mode is selected by pressing the Debug on/off button. When the debug mode is on, an additional set of buttons are displayed. This screen is shown in Figure 9. This mode allows the user to set breakpoints, view opcodes, view memory, and step through machine code. In addition to the default mode of operation (Figure 8) the following buttons are displayed:

- Toggle Step Mode/Run Mode
- Single Step
- Set Breakpoint
- Generate Interrupt (NMI)

- Dump Memory
- Options

2.2.8.6 Step and Run Mode

The Step Mode/Run Mode button toggles between stepping manually through the emulation and running the emulation at full speed. When in step mode, the user may single-step through the emulation by pressing the step button. One instruction is executed each time the step button is pressed.

When in run mode, the step button becomes inactive, and the system is run at full speed.

2.2.8.7 Set Breakpoint

The Set Breakpoint button shall prompt the user at what location in memory the user wants the emulation to stop executing and set the Step Mode On.

2.2.8.8 Generate NMI Interrupt

The Generate NMI button shall generate a non-maskable interrupt. This is done to test the interrupt handling of the emulator. The CPU NMI line is activated when this button is pressed.

2.2.8.9 Memory Dump

The Memory Dump button prompts the user for a memory area to display. A separate window displays the area selected.

- The memory dump display shall be within a separate window
- The memory dump shall display in hexadecimal
- The memory dump shall include addresses
- The memory dump window can remain displayed, minimized, or closed.

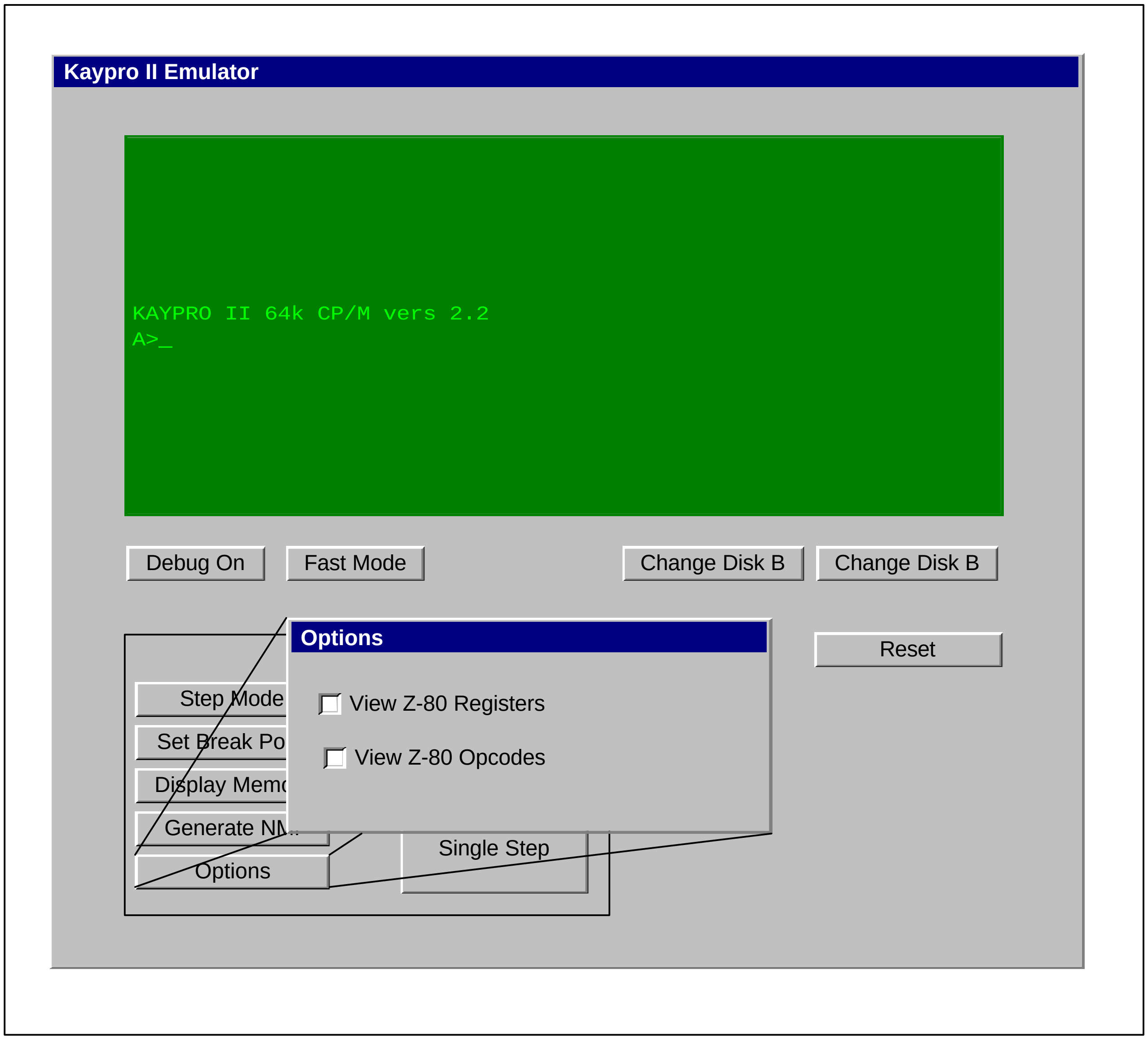
2.2.8.10 View options

When single-stepping through the emulation, the user may wish to view the opcodes and/or CPU flags. Because the display screen size may be limited, the user may wish to eliminate display of either flags or opcodes. This shall be accomplished via the View Options menu. An example screen is shown in Figure 10.

When the View Options button is pressed a dialog box shall appear. The View Options dialog shall allow the user to enable or disable display of opcodes or CPU flags.

The default action shall be:

- Enable flags
- Enable opcode display.



```
KAYPRO II 64k CP/M vers 2.2
A>_
```

Figure 10, View Options

2.2.9 Devices

As mentioned before, many devices map to specific ports. Those devices are:

- Floppy drive
- Serial I/O
- Keyboard/Beep
- Parallel/Printer
- Baud Rate
- Misc. control

2.2.9.1 Serial I/O

The serial port in the Kaypro II is based on the 3884 SIO (Serial Input/output). The 3884 is a dual channel serial communication chip. It combines two serial ports on a single chip. Serial port A is utilized as a standard external serial input/output. Port B is used to communicate with the keyboard and system beep signal.

2.2.9.1.1 Summary

The SIO obtains timing from a dual baud rate generator (described below). The SIO communicates with the Z-80 via a control register and a data register pair.

- The SIO can be set in asynchronous, or synchronous modes.
- The SIO is capable of implementing interrupt-driven functions.
- The SIO utilizes maskable interrupts (the INT line on the Z-80 CPU)
- A single interrupt line is utilized for channel A and channel B serial devices.

2.2.9.1.2 Interrupts

The SIO is capable of daisy chaining interrupts. This allows it to coordinate function with other interrupt driven devices. The SIO is the highest priority device (see interrupts).
The SIO shall have priority over all other devices if it attempts to assert an interrupt.

2.2.9.1.3 Registers and Addressing

The SIO chip is programmed via four registers, two for each channel. These registers are shown in Table 5.

Register Name	Function	Port
SIO Channel A Data	Passes data to and from SIO A	0x04
SIO Channel B Data	Passes data to and from SIO A	0x05
SIO Channel A Control	Used as further index to internal register	0x06
SIO Channel B Control	Used as further index to internal register	0x07

Table 5, SIO Interface Registers

The SIO interface registers act as an indirect address to a set of eight internal registers. The Z-80 selects an internal register to address via an output to an associated control channel. Reading or writing to a specific internal register is a two-step process. First, the register is selected (or indexed) via the associated control register. Next, the data is read or written to/from the port via the associated SIO data register.

Figure 11 shows an example of this addressing scheme. Here, the Z-80 delivers an index to the SIO chip via the SIO control port. Next, the index is used to reference into the internal SIO register set. Data can then be read/written to the indexed internal register. Until the index within the SIO is re-written, or the SIO is reset, the index value remains. That means that subsequent data reads and writes need not re-specify an index if it already exists.

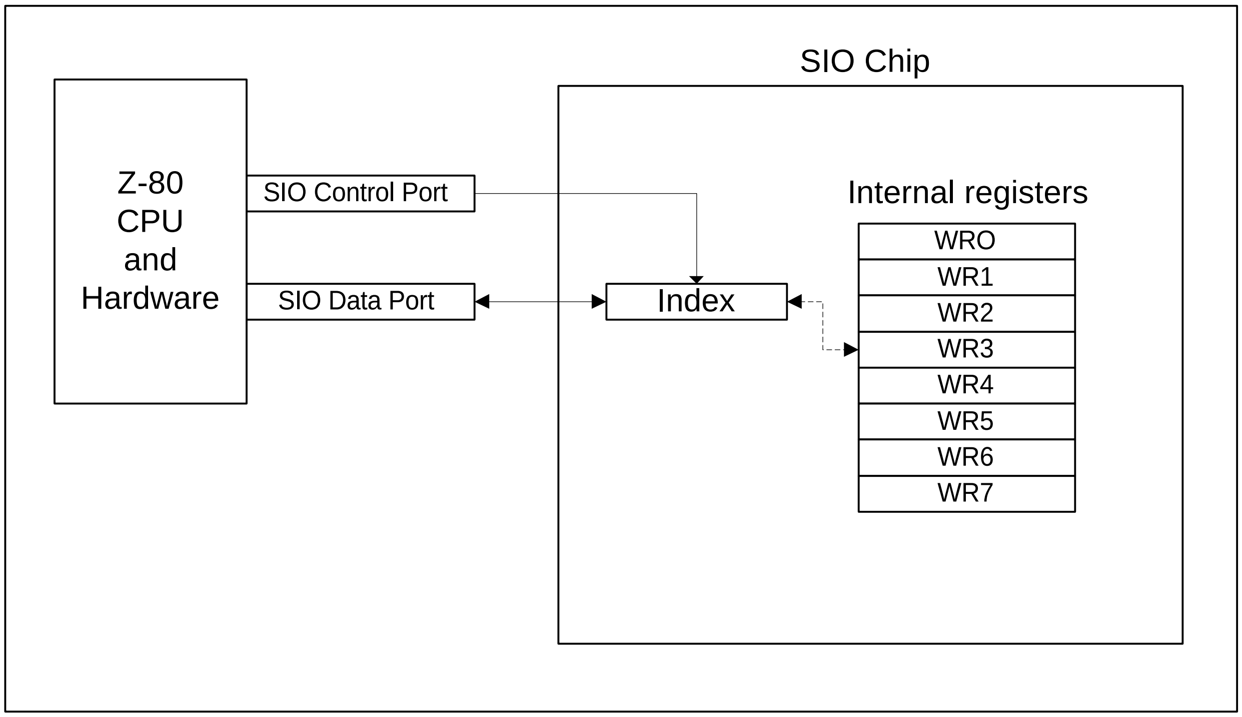


Figure 11, SIO Indexing Scheme

The SIO contains 8 internal registers. The registers may be read from or written to. Register 0 is unique, in that it duplicates the function of the SIO control register (Figure 11, SIO Indexing Scheme).

Each register has a set of unique functions. The registers may contain data parameters, return data parameters or status, or serve as command registers. These registers are shown in Table 6 and Table 7.

- The SIO chip channel B provides for keyboard input and speaker output. These functions shall be implemented on a function level. That is, specific configuration of the SIO channel A shall be ignored, except where it influences functional operation.
- SIO baud rate may be programmed via command and data registers. The Kaypro II emulation shall accept all values for baud rate, parity, start word, and stop words, but shall operate at system speed. That is, these values shall be ignored.
- Synchronous mode shall not be supported.

Write Register	Summary	Commands and Status
WR0	Command Register, CRC reset and register pointer	- Null Command - Reset external/status interrupt - Channel reset - Enable interrupt on next receive character - Reset the transmitter due to pending interrupt - Error reset - Return from interrupt.
WR1	Interrupt enable and wait/ready modes	- External/status interrupt enable - Transmitter interrupt enable - Status affects for ISR vector. - Transmitter interrupt disable - Receive interrupts disabled. - Receive interrupt on first char only - Receive interrupts enabled, parity error Special Receive - Receive interrupts enabled, parity err non-special
WR2	Interrupt vector address pointer	**Associated Values Normal Condition:** - Half Address pointer to ISR. **Associated Values if WR1 has Status Affects Vector value** - Channel b transmit buffer empty - Channel b external/ status change - Channel b receive character available - Channel b special receive condition (Parity error, RD overrun).
WR3	Receiver logic control and parameters	- Receiver enable - Auto enable - 5 bit per character transfer - 6 bit per character transfer - 7 bit per character transfer - 8 bit per character transfer
WR4	Control bits for use with transmit and receive	- Enable Parity - Parity even - Sync command on. - 1 stop bit - 1.5 stop bits. - 2 stop bits. - The emulated WR4 shall not support x clock rate stepping.
WR5	Control bits for transmission	- Transmit enable - Send break - Bits per char to transmit 5,7,6,8 - RTS output - DTR output
WR6	SDLC transmit synchronization character	Storage for SDLC transmit synchronization character
WR7	SDLC receive synchronization character	Storage for SDLC receive synchronization character

Table 6, SIO Internal (Write) Registers

Read Register	Summary	Commands and Status
0	General receive and transmit status	- Receive character available - Interrupt pending - Transmit buffer empty - sync/hunt - DCD input - CTS input - Transmit under-run end of memory - Break/ abort status
1	Special receive conditions and residue codes	- Parity error - Rx overrun error - Framing error.
2	Interrupt vector address/pointer (channel B only)	- Interrupt vector address/pointer
3	N/A	N/A
4	N/A	N/A
5	N/A	N/A
6	N/A	N/A
7	N/A	N/A

Table 7, SIO Internal (Read Registers)

2.2.9.2 Floppy Emulation

2.2.9.2.1 Summary

The Kaypro II included two 5 ¼" floppy disk drives. Each floppy was capable of holding up to 195K bytes of information. Typically the CP/M operating system took 4K, leaving 191K bytes for user storage. The floppy disks specifications were as follows.

- Double density
- 40 tracks per diskette
- 10 sectors per track
- 512 bytes per sector

Disk drive control shall be emulated at the hardware level with virtual machine translation for a non-CP/M, DOS formatted diskettes, or RAM drives holding a virtual image of the CP/M operating system. This will allow the maximum compatibility between actual media, and emulated media.

- The emulation shall contain two virtual floppy disk drives.
- Electronic copies of actual disks shall be obtained and translated into emulator format
- Disk data shall be copies of individual sector data
- Track and sector numbering shall be maintained
- Data alignment shall be maintained
- All unused bytes contained within the virtual floppy shall contain the value: 0xE5.

2.2.9.2.2 1792 Floppy Controller (FDC)

The 1791 floppy controller chip from Synertek was used to control the floppy hardware. It provides a programmatic interface between the software and the physical floppy transport.

2.2.9.2.3 Interrupts

- NMI interrupts shall be supported as generated by the 1791 floppy disk controller.
- The 1791 generates NMI interrupts (see interrupts). These NMI interrupts indicate when the 1791 has read or written a single byte of data to the floppy diskette.
- The 1791 generates an NMI interrupt at the completion of each command
- The 1791 generates two NMI interrupts when a command and data operation is requested at the same time (one for command completion, and one for read/write of data).

2.2.9.2.4 Registers and Addressing

The FDC chip is programmed via six registers. These registers are shown in Table 8 and Table 9.

Read Register	Summary	Commands and Status	Port
Command/status register	Holds current command/status for 1791 FDC	Read current status	0x10
Track register	Holds current track being read	Read current track selection	0x11
Sector register	Holds current sector being read	Read current sector selection	0x12
Data	Holds data that was read from floppy	Read data stored on floppy disk, a byte at a time (for Kaypro II)	0x13

Table 8, FDC Registers (Read)

Write Register	Summary	Commands and Status	Port
Command/status register	Holds current command for 1791 FDC	• Restore • Seek • Step (step in same direction as last command) • STEP IN (STEP IN TRACK+1) • Step out (step out track-1) • Read sector (execute a read sector) • Write sector (execute a write sector) • Read address (read next ID field) • Read track (executes a read track) • Write track (executes a write track) • Force interrupt (force termination of current command)	0x10
Track register	Holds current track to write	Write current track selection	0x11
Sector register	Holds current sector to write	Write current sector selection	0x12
Data	Holds data to be written to floppy	Present data to be stored on floppy disk, a byte at a time (for Kaypro II)	0x13

Table 9, FDC Registers (Write)

2.2.9.2.5 FDC Read Write Scenario

The FDC commands are typically used to read and write data to and from a floppy diskette. The Kaypro II executes the following sequence when reading or writing diskette (typically a sector read).

1. Send track to be read/written to track register
2. NMI marks end of command
3. Send sector to be read/written to sector register
4. NMI marks end of command
5. Execute a read or write command
6. NMI marks end of command
7. CPU executes a HALT
8. FDC pulls NMI line, indicating one byte of data retrieval
9. CPU wakes from halt when it receives the NMI
10. CPU reads from data port
11. If all data is read, it exits read/write loop
12. Else, it goes back to step 7

2.2.9.2.6 Other Floppy Operations

Other floppy operations include:

- Disk motor control
- Floppy drive select (A or B)
- Floppy side select (side of dual density floppy to read)

These operations are handled by the PIO port (below).

2.2.9.3 Keyboard Functions

The keyboard functions are supported through the second serial port. All keystrokes are handled by SIO port B. This port may not be configured any other way. It is important to emulate this functionality as closely as possible. The user will type directly into the emulation. That is, there will be no external windows for entry into the system.

- Keyboard input shall be made directly into the Java page. The user shall be able to type directly into the emulated Kaypro II
- The keyboard input shall be emulated via the B SIO port

2.2.9.4 Parallel/Printer Port

The parallel ports in the Kaypro II are based on two 3881 PIO's (Parallel Input/output). The 3881 PIO is a dual channel parallel communication controller. It combines two parallel interfaces on a single chip (A and B). Unlike the serial chip, only one parallel interface is utilized per chip (A). The Kaypro II utilizes two parallel interfaces. That means that two separate PIO chips were required. Perhaps this is because only one interface on each dual controller supports bi-directional data flow.

Parallel interface 1 is used as a standard externally accessible parallel port. This interface terminates on the back of the Kaypro II cabinet. It is typically used to connect a Centronics compatible printer to the Kaypro II. Parallel interface 2 is used to communicate with the internal system functions. It is used to drive various devices within the Kaypro II itself. These devices are described below.

2.2.9.4.1 Summary

The PIO pair (1 and 2) communicate with the Z-80 via a control register and a data register pair.

- The PIO is capable of byte, I/O, bi-directional operation, and bit control mode.
- The PIO is capable of implementing interrupt-driven functions.
- The PIO utilizes maskable interrupts (the INT line on the Z-80 CPU)

- A single interrupt line is utilized for all parallel serial devices.

2.2.9.4.2 Interrupts

The PIO is capable of daisy chaining interrupts. This allows it to coordinate function with other interrupt driven devices. The two PIO's are the lowest priority devices (see interrupts).

- The PIO port 1 (external connection) shall have the lowest priority
- PIO port 2 (internal operations) shall have a higher priority than port 1, but less than SIO
- The PIO shall communicate utilizing mode 2 interrupts

2.2.9.4.3 Registers and Addressing

The PIO chip is programmed via four registers, two for each channel. These registers are shown in Table 10, PIO Interface Registers.

Register Name	Function	Port
PIO 1 Channel A Data	Passes data to and from PIO A (printer interface)	0x08
PIO 1 Channel A Control	Used as further index to internal register (printer interface)	0x09
PIO 1 Channel B Data	Not used	0x0A
PIO 1 Channel B Control	Not Used	0x0B
PIO 2 Channel A Data	Passes data to and from PIO A (internal device interface)	0x1C
PIO 2 Channel A Control	Used as further index to internal register (internal device interface)	0x1D
PIO 2 Channel B Data	Not Used	0x1E
PIO 2 Channel B Control	Not Used	0x1F

Table 10, PIO Interface Registers

The control registers send commands to the PIO itself. These registers are used to program the operation of the PIO. The data registers are used to read and write data to/from the PIO. This is illustrated in Table 11.

Register	Command	Actions
Control	Load interrupt vector	- Load interrupt vector into PIO
Control	Set interrupt control word	- Enable/disable interrupts - And/OR - High/Low - Mask follows
Control	Select operating mode	- Output - Input - Bi-directional - Control
Data	NA	Read or write data to/from PIO

Table 11, PIO Register Functions

- The PIO shall support only output mode. The Kaypro II is hard-wired for printer operation.
- PIO 1 shall act as the printer port. It shall accept output via its data channel, and print the data to a separate window as ASCII text. The size of the printer buffer shall be limited only by the Java control used.
- Configuration commands sent to PIO 1 shall be accepted but ignored. Instead, PIO 1 shall always function as a printer port.
- PIO 2 shall function as an interface to internal and external devices. Although output to these devices shall be supported, some may not be needed for emulation operation. These devices include:
    - Disk motor control
    - Printer strobe
    - Printer busy
    - Floppy drive select (A or B)
    - Floppy side select (side of dual density floppy to read or write)

2.2.9.5 Baud Rate Port

The 8116 dual baud rate chip used in the Kaypro II shall accept output from the Z-80 CPU. Typically the baud rates supported by the Kaypro II are:

- 110 baud
- 300 baud
- 1200 baud
- 2400 baud
- 4800 baud
- 9600 baud
- 19.2k baud

- These baud rates shall be accepted by the emulation. However, their values will be disregarded.
- The SIO shall operate at system speed (as fast as the environment will allow).

The port definitions for the baud rate generators are specified in Table 12.

Port	Usage
0x00	Baud rate generator A
0x0C	Baud rate generator B

Table 12, Baud Rate Port Assignments

2.2.10 Miscellaneous Ports

- The emulator shall not crash due to writes to unsupported functionality. Specifically reads or writes to serial and parallel ports

2.2.11 Bootstrap Loader

The Kaypro II utilizes a unique way of loading the CP/M operating system. Older systems required manually loading the bootstrap code.

The Kaypro II. Uses an internal ROM. This ROM contains the startup code needed to bring CP/M into memory.

Typical CP/M diskettes contained a short bootstrap program on the lowest track and sector. The Kaypro II stores the location to load CP/M and length of the CP/M operating system in this area instead. This should be noted. This should not be a problem for the emulator if the CP/M floppy diskettes are faithfully duplicated.

2.2.12 Operating system

The operating system shall be supported on disk images. The disk images shall contain CP/M 2.2. The operating system shall be read from a valid Kaypro II diskette, and transferred electronically into a form recognizable by the Kaypro II emulator. Once inside the emulator, the disk images shall be loaded via one of two virtual floppy disk drives.

- CP/M 2.2 shall be supported
- CP/M OS shall be supplied on track 1 of each virtual floppy disk.
- The emulator shall support loading of the OS from floppy drive A
- The CP/M operating system shall actually be run at the software level via an obtained copy of the CP/M operating system

3. Project Deliverables

This section identifies all deliverable components of the project including hardware, software, training, and documentation.

3.1 Hardware

No hardware shall be delivered

3.2 Software

All Kaypro II software shall be delivered.
All source code shall be delivered.
All associated build or make files shall be delivered

3.3 Training

No training shall be provided.

3.4 Project Documentation

There are two categories of documents: project development documents, such as the project plan and design specification, and customer documents, such as the user's guide. These documents are delivered according to the project schedule.

3.4.1 Project Development Documentation

Requirements design documents shall be provided
Requirements specifications document shall be provided
Design documents shall be provided

3.4.2 Customer/Operations Documentation

A user guide shall be provided

4. Applicable Documents, Reference, and Glossary

This section contains title, author, and publication information for documents referred to or having an impact on the requirements for this project. It also contains a comprehensive glossary of applicable terms and acronyms.

4.1 References

Requirements Definition For The CSI426/Kaypro II Emulator
Zilog Z80 Microprocessor Family User's Manual, Part number Q1/95 DC 8309-1
*Z80.DOC, opcode reference, compiled by Sean Young (*syoung@cs.vu.nl*)*
Synertek Data Book, 1983

4.2 Appendix A, Z-80 Opcodes

4.2.1 8 bit Load Group

Mnemonic	Symbolic Operation	Flags S Z F5 H F3 P/V N C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD r, r'	r ← r'	• • • • • • • • •	01 r r'		1	1	4	r, r' Reg.
LD p, p'*	p ← p'	• • • • • • • • •	11 011 101	DD	2	2	8	000 B
			01 p p'					001 C
LD q, q'*	q ← q'	• • • • • • • • •	11 111 101	FD	2	2	8	010 D
			01 q q'					011 E
LD r, n	r ← n	• • • • • • • • •	00 r 110		2	2	7	100 H
			← n →					101 L
LD p, n*	p ← n	• • • • • • • • •	11 011 101	DD	3	3	11	111 A
			00 p 110					
			← n →					p, p' Reg.
LD q, n*	q ← n	• • • • • • • • •	11 111 101	FD	3	3	11	000 B
			00 q 110					001 C
			← n →					010 D
LD r, (HL)	r ← (HL)	• • • • • • • • •	01 r 110		1	2	7	011 E
LD r, (IX + d)	r ← (IX + d)	• • • • • • • • •	11 011 101	DD	3	5	19	100 IX_H
			01 r 110					101 IX_L
			← d →					111 A
LD r, (IY + d)	r ← (IY + d)	• • • • • • • • •	11 111 101	FD	3	5	19	
			01 r 110					q, q' Reg.
			← d →					000 B
LD (HL), r	(HL) ← r	• • • • • • • • •	01 110 r		1	2	7	001 C
LD (IX + d), r	(IX + d) ← r	• • • • • • • • •	11 011 101	DD	3	5	19	010 D
			01 110 r					011 E
			← d →					100 IY_H
LD (IY + d), r	(IY + d) ← r	• • • • • • • • •	11 111 101	FD	3	5	19	101 IY_L
			01 110 r					111 A
			← d →					
LD (HL), n	(HL) ← n	• • • • • • • • •	00 110 110	36	2	3	10	
			← n →					
LD (IX + d), n	(IX + d) ← n	• • • • • • • • •	11 011 101	DD	4	5	19	
			00 110 110	36				
			← d →					
			← n →					
LD (IY + d), n	(IY + d) ← n	• • • • • • • • •	11 111 101	FD	4	5	19	
			00 110 110	36				
			← d →					
			← n →					
LD A, (BC)	A ← (BC)	• • • • • • • • •	00 001 010	0A	1	2	7	
LD A, (DE)	A ← (DE)	• • • • • • • • •	00 011 010	1A	1	2	7	
LD A, (nn)	A ← (nn)	• • • • • • • • •	00 111 010	3A	3	4	13	
			← n →					
			← n →					
LD (BC), A	(BC) ← A	• • • • • • • • •	00 000 010	02	1	2	7	
LD (DE), A	(DE) ← A	• • • • • • • • •	00 010 010	12	1	2	7	
LD (nn), A	(nn) ← A	• • • • • • • • •	00 110 010	32	3	4	13	
			← n →					
			← n →					
LD A, I	A ← I	↕ ↕ ↕ 0 ↕ IFF_2 0 •	11 101 101	ED	2	2	9	
			01 010 111	57				
LD A, R	A ← R	↕ ↕ ↕ 0 ↕ IFF_2 0 •	11 101 101	ED	2	2	9	R is read after it
			01 011 111	5F				is increased.
LD I, A	I ← A	• • • • • • • • •	11 101 101	ED	2	2	9	
			01 000 111	47				
LD R, A	R ← A	• • • • • • • • •	11 101 101	ED	2	2	9	R is written after i
			01 001 111	4F				is increased.

Notes: r, r' means any of the registers A, B, C, D, E, H, L.
p, p' means any of the registers A, B, C, D, E, IX_H, IX_L.
q, q' means any of the registers A, B, C, D, E, IY_H, IY_L.
dd_L, dd_H refer to high order and low order eight bits of the register respectively.

	* means unofficial instruction.
Flag Notation:	• = flag is not affected, 0 = flag is reset, 1 = flag is set,
	↕ = flag is set according to the result of the operation, IFF$_2$ = the interrupt flip-flop 2 is copied.

4.2.2 16 bit Load Group

	Symbolic	Flags								Opcode		No. of	No. of M	No. of T	
Mnemonic	Operation	S	Z	F5	H	F3	P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments
LD dd, nn	dd ← nn	•	•	•	•	•	•	•	•	00 dd0 001		3	3	10	dd Pair
										← n →					00 BC
										← n →					01 DE
LD IX, nn	IX ← nn	•	•	•	•	•	•	•	•	11 011 101	DD	4	4	14	02 HL
										00 110 001	21				03 SP
										← n →					
										← n →					
LD IY, nn	IY ← nn	•	•	•	•	•	•	•	•	11 111 101	FD	4	4	14	
										00 110 001	21				
										← n →					
										← n →					
LD HL, (nn)	L ← (nn)	•	•	•	•	•	•	•	•	00 101 010	2A	3	5	16	
	H ← (nn+1)									← n →					
										← n →					
LD dd, (nn)	dd_L ← (nn)	•	•	•	•	•	•	•	•	11 101 101	ED	4	6	20	
	dd_H ←									01 dd1 011					
	(nn+1)									← n →					
										← n →					
LD IX, (nn)	IX_L ← (nn)	•	•	•	•	•	•	•	•	11 011 101	DD	4	6	20	
	IX_H ← (nn+1)									00 101 010	2A				
										← n →					
										← n →					
LD IY, (nn)	IY_L ← (nn)	•	•	•	•	•	•	•	•	11 111 101	FD	4	6	20	
	IY_H ← (nn+1)									00 101 010	2A				
										← n →					
										← n →					
LD (nn), HL	(nn) ← L	•	•	•	•	•	•	•	•	00 100 010	22	3	5	16	
	(nn+1) ← H									← n →					
										← n →					
LD (nn), dd	(nn) ← dd_L	•	•	•	•	•	•	•	•	11 101 101	DD	4	6	20	
	(nn+1) ←									01 dd0 011					
	dd_H									← n →					
										← n →					
LD (nn), IX	(nn) ← IX_L	•	•	•	•	•	•	•	•	11 011 101	DD	4	6	20	
	(nn+1) ← IX_H									00 100 010	22				
										← n →					
										← n →					
LD (nn), IY	(nn) ← IY_L	•	•	•	•	•	•	•	•	11 111 101	FD	4	6	20	
	(nn+1) ← IY_H									00 100 010	22				
										← n →					
										← n →					
LD SP, HL	SP ← HL	•	•	•	•	•	•	•	•	11 111 001	F9	1	1	6	
LD SP, IX	SP ← IX	•	•	•	•	•	•	•	•	11 011 101	DD	2	2	10	
										11 111 001	F9				
LD SP, IY	SP ← IY	•	•	•	•	•	•	•	•	11 111 101	FD	2	2	10	
										11 111 001	F9				
PUSH qq	SP ← SP - 1	•	•	•	•	•	•	•	•	11 qq0 101		1	3	11	qq Pair
	(SP) ← qq_H														00 BC
	SP ← SP - 1														01 DE
	(SP) ← qq_L														10 HL
PUSH IX	SP ← SP - 1	•	•	•	•	•	•	•	•	11 011 101	DD	2	4	15	11 AF
	(SP) ← IX_H									11 100 101	E5				
	SP ← SP - 1														
	(SP) ← IX_L														
PUSH IY	SP ← SP - 1	•	•	•	•	•	•	•	•	11 111 101	FD	2	4	15	
	(SP) ← IY_H									11 100 101	E5				
	SP ← SP - 1														
	(SP) ← IY_L														

POP qq	(SP) ← qq_L SP ← SP + 1 (SP) ← qq_H SP ← SP + 1	•	•	•	•	•	•	•	•	11 qq0 001		1	3	10
POP IX	(SP) ← IX_L SP ← SP + 1 (SP) ← IX_H SP ← SP + 1	•	•	•	•	•	•	•	•	11 011 101 11 100 001	DD E1	2	4	14
POP IY	(SP) ← IY_L SP ← SP + 1 (SP) ← IY_H SP ← SP + 1	•	•	•	•	•	•	•	•	11 111 101 11 100 001	FD E1	2	4	14

Notes:	dd is any of the register pair BC, DE, HL, SP. qq is any of the register pair BC, DE, HL, AF.
Flag Notation:	• = flag is not affected, 0 = flag is reset, 1 = flag is set, ↕ = flag is set according to the result of the operation.

4.2.3 Exchange, Block Transfer and Search Groups

Mnemonic	Symbolic Operation	S	Z	F5	H	F3	P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
EX DE, HL	DE ↔ HL	•	•	•	•	•	•	•	•	11 101 011	EB	1	1	4	
EX AF, AF’	AF ↔ AF’	•	•	•	•	•	•	•	•	00 001 000	08	1	1	4	
EXX	BC ↔ BC’ DE ↔ DE’ HL ↔ HL’	•	•	•	•	•	•	•	•	11 011 001	D9	1	1	4	
EX (SP), HL	(SP+1) ↔ H (SP) ↔ L	•	•	•	•	•	•	•	•	11 100 011	E3	1	5	19	
EX (SP), IX	(SP+1) ↔ IX_H (SP) ↔ IX_L	•	•	•	•	•	•	•	•	11 011 101 11 100 011	DD E3	2	6	23	
EX (SP), IY	(SP+1) ↔ IY_H (SP) ↔ IY_L	•	•	•	•	•	•	•	•	11 111 101 11 100 011	FD E3	2	6	23	
LDI	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC - 1	•	•	↕[1]	0	↕[2]	↕[3]	0	•	11 101 101 10 100 000	ED A0	2	4	16	
LDIR	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC - 1 repeat until: BC = 0	•	•	↕[1]	0	↕[2]	0	0	•	11 101 101 10 110 000	ED B0	2 2	5 4	21 16	if BC ≠ 0 if BC = 0
LDD	(DE) ← (HL) DE ← DE - 1 HL ← HL - 1 BC ← BC - 1	•	•	↕[1]	0	↕[2]	↕[3]	0	•	11 101 101 10 101 000	ED A8	2	4	16	
LDDR	(DE) ← (HL) DE ← DE - 1 HL ← HL - 1 BC ← BC - 1 repeat until: BC = 0	•	•	↕[1]	0	↕[2]	0	0	•	11 101 101 10 111 000	ED B8	2 2	5 4	21 16	if BC ≠ 0 if BC = 0
CPI	A - (HL) HL ← HL + 1 BC ← BC -1	↕[4]	↕[4]	↕[5]	↕[4]	↕[6]	↕[3]	1	•	11 101 101 10 100 001	ED A1	2	4	16	
CPIR	A - (HL) HL ← HL + 1 BC ← BC -1 Repeat until: A = (HL) or BC = 0	↕[4]	↕[4]	↕[5]	↕[4]	↕[6]	↕[3]	1	•	11 101 101 10 110 001	ED B1	2 2	5 4	21 16	if BC ≠ 0 and A ≠ (HL). if BC = 0 or A = (HL)
CPD	A - (HL) HL ← HL - 1 BC ← BC -1	↕[4]	↕[4]	↕[5]	↕[4]	↕[6]	↕[3]	1	•	11 101 101 10 101 001	ED A9	2	4	16	
CPDR	A - (HL) HL ← HL - 1 BC ← BC -1 Repeat until: A = (HL) or BC = 0	↕[4]	↕[4]	↕[5]	↕[4]	↕[6]	↕[3]	1	•	11 101 101 10 111 001	ED B9	2 2	5 4	21 16	if BC ≠ 0 and A ≠ (HL). if BC = 0 or A = (HL)

Notes:
[1] F5 is a copy of bit 1 of A + last transferred byte, thus $(A + (HL))_1$
[2] F3 is a copy of bit 3 of A + last transferred byte, thus $(A + (HL))_3$
[3] P/V flag is 0 if the result of BC - 1 = 0, otherwise P/V = 1.
[4] These flags are set as in CP (HL)
[5] F5 is copy of bit 1 of A - last compared address - H, thus $(A - (HL) - H)_1$. H is as in F after the comparison.
[6] F3 is copy of bit 3 of A - last compared address - H, thus $(A - (HL) - H)_3$. H is as in F after the comparison.

Flag Notation: • = flag is not affected, 0 = flag is reset, 1 = flag is set, ↕ = flag is set according to the result of the operation.

4.2.4 8 bit Arithmetic and Logical Group

Mnemonic	Symbolic Operation	S	Z	F5	H	F3	P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
ADD A, r	A ← A + r	↕	↕	↕	↕	↕	V	0	↕	10 000 r		1	1	4	r Reg. p Re
ADD A, p*	A ← A + p	↕	↕	↕	↕	↕	V	0	↕	11 011 101 10 000 p	DD	2	2	8	000 B 000 B 001 C 001 C
ADD A, q*	A ← A + q	↕	↕	↕	↕	↕	V	0	↕	11 111 101 10 000 q	FD	2	2	8	010 D 010 D 011 E 011 E
ADD A, n	A ← A + n	↕	↕	↕	↕	↕	V	0	↕	11 000 110 ← n →		2	2	8	100 H 100 IX 101 L 101 IX
ADD A, (HL)	A ← A + (HL)	↕	↕	↕	↕	↕	V	0	↕	10 000 110		1	2	7	111 A 111 A
ADD A, (IX + d)	A ← A + (IX + d)	↕	↕	↕	↕	↕	V	0	↕	11 011 101 10 000 110 ← d →	DD	3	5	19	
ADD A, (IY + d)	A ← A + (IY + d)	↕	↕	↕	↕	↕	V	0	↕	11 111 101 10 000 110 ← d →	FD	3	5	19	
ADC A, s	A ← A + s + CY	↕	↕	↕	↕	↕	V	0	↕	001					s is any of r, n, (HL)
SUB A, s	A ← A - s	↕	↕	↕	↕	↕	V	1	↕	010					(IX+d), (IY+d), p, q
SBC A, s	A ← A - s - CY	↕	↕	↕	↕	↕	V	1	↕	011					as shown for the AD
AND s	A ← A AND s	↕	↕	↕	1	↕	P	0	0	100					instruction. The
OR s	A ← A OR s	↕	↕	↕	0	↕	P	0	0	110					underlined bits replace
XOR s	A ← A XOR s	↕	↕	↕	0	↕	P	0	0	101					the underlined bits i
CP s	A - s	↕	↕	↕[1]	↕	↕[1]	V	1	↕	111					the ADD set.
INC r	r ← r + 1	↕	↕	↕	↕	↕	V	0	•	00 r 100		1	1	4	
INC p*	p ← p + 1	↕	↕	↕	↕	↕	V	0	•	11 011 101 00 p 100	DD	2	2	8	q Reg. 000 B
INC q*	q ← q + 1	↕	↕	↕	↕	↕	V	0	•	11 111 101 00 q 100	FD	2	2	8	001 C 010 D
INC (HL)	(HL) ← (HL) + 1	↕	↕	↕	↕	↕	V	0	•	00 110 100		1	3	11	011 E
INC (IX + d)	(IX + d) ← (IX + d) + 1	↕	↕	↕	↕	↕	V	0	•	11 011 101 00 110 100 ← d →	DD	3	6	23	100 IY_H 101 IY_L 111 A
INC (IY + d)	(IY + d) ← (IY + d) + 1	↕	↕	↕	↕	↕	V	0	•	11 111 101 00 110 100 ← d →	FD	3	6	23	
DEC m	m ← m - 1	↕	↕	↕	↕	↕	V	1	•	101					m is any of r, p, q, (HL), (IX+d), (IY+d), as shown for the IN instruction. DEC same format and states as INC. Replace 100 with 1(in opcode.

Notes: [1] F5 and F3 are copied from the operand (s), not from the result of (A - s).
The V symbol in the P/V flag column indicates that the P/V flags contains the overflow of the operation. Similarly the P symbol indicates parity.
r means any of the registers A, B, C, D, E, H, L.
p means any of the registers A, B, C, D, E, IX_H, IX_L.
q means any of the registers A, B, C, D, E, IY_H, IY_L.
dd_L, dd_H refer to high order and low order eight bits of the register respectively.
CY means the carry flip-flop.
* means unofficial instruction.

Flag Notation: • = flag is not affected, 0 = flag is reset, 1 = flag is set, ↕ = flag is set according to the result of the operation.

4.2.5 16 bit Arithmetic Group

	Symbolic	Flags								Opcode		No.of	No.of M	No.of T	
Mnemonic	Operation	S	Z	F5	H	F3	P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments
ADD HL, ss	HL ← HL + ss	•	•	$\updownarrow^2$	$\updownarrow^2$	$\updownarrow^2$	•	0	$\updownarrow^1$	00 ss1 001		1	3	11	ss Reg.
ADC HL, ss	HL ← HL + ss + CY	$\updownarrow^1$	$\updownarrow^1$	$\updownarrow^2$	$\updownarrow^2$	$\updownarrow^2$	V^1	0	$\updownarrow^1$	11 101 101	ED	2	4	15	00 BC
										01 ss1 010					01 DE
SBC HL, ss	HL ← HL - ss - CY	$\updownarrow^1$	$\updownarrow^1$	$\updownarrow^2$	$\updownarrow^2$	$\updownarrow^2$	V^1	1	$\updownarrow^1$	11 101 101	ED	2	4	15	10 HL
										01 ss0 010					11 SP
ADD IX, pp	IX ← IX + pp	•	•	$\updownarrow^2$	$\updownarrow^2$	$\updownarrow^2$	•	0	$\updownarrow^1$	11 011 101	DD	2	4	15	
										00 pp1 001					pp Reg.
ADD IY, rr	IY ← IY + rr	•	•	$\updownarrow^2$	$\updownarrow^2$	$\updownarrow^2$	•	0	$\updownarrow^1$	11 111 101	FD	2	4	15	00 BC
										00 rr1 001					01 DE
INC ss	ss ← ss + 1	•	•	•	•	•	•	•	•	00 ss0 011		1	1	6	10 IX
INC IX	IX ← IX + 1	•	•	•	•	•	•	•	•	11 011 101	DD	2	2	10	11 SP
										00 100 011	23				
INC IY	IY ← IY + 1	•	•	•	•	•	•	•	•	11 111 101	FD	2	2	10	rr Reg.
										00 100 011	23				00 BC
DEC ss	ss ← ss - 1	•	•	•	•	•	•	•	•	00 ss1 011		1	1	6	01 DE
DEC IX	IX ← IX - 1	•	•	•	•	•	•	•	•	11 011 101	DD	2	2	10	10 IY
										00 101 011	2B				11 SP
DEC IY	IY ← IY - 1	•	•	•	•	•	•	•	•	11 111 101	FD	2	2	10	
										00 101 011	2B				

Notes: The V symbol in the P/V flag column indicates that the P/V flags contains the overflow of the operation.
ss means any of the registers BC, DE, HL, SP.
pp means any of the registers BC, DE, IX, SP.
rr means any of the registers BC, DE, IY, SP.
16 bit additions are performed by first adding the two low order eight bits, and then the two high order eight bits.
[1] Indicates the flag is affected by the 16 bit result of the operation.
[2] Indicates the flag is affected by the 8 bit addition of the high order eight bits.
CY means the carry flip-flop.

Flag Notation: • = flag is not affected, 0 = flag is reset, 1 = flag is set, ↕ = flag is set according to the result of the operation.

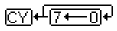
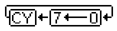
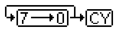
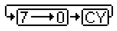
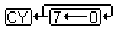
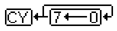
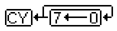
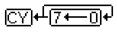
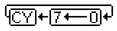
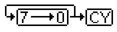
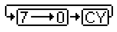
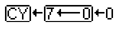
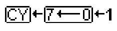
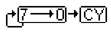
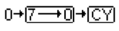
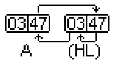
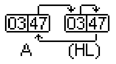
4.2.6 General Purpose Arithmetic and CPU Control Groups

Mnemonic	Symbolic Operation	Flags S	Z	F5	H	F3	P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
DAA	Converts A into packed BCD following add or subtract with BCD operands.	↕	↕	↕	↕	↕	P	•	↕	00 100 111	27	1	1	4	
CPL	A ← $\overline{A}$	•	•	↕[1]	1	↕[1]	•	1	•	00 101 111	2F	1	1	4	One's complement.
NEG[4]	A ← 0 - A	↕	↕	↕	↕	↕	V	1	↕	11 101 101 01 000 100	ED 44	2	2	8	Two's complement.
CCF	CY ← $\overline{CY}$	•	•	↕[1]	↕[2]	↕[1]	•	0	↕	00 111 111	3F	1	1	4	Complement carry flag.
SCF	CY ← 1	•	•	↕[1]	0	↕[1]	•	0	1	00 110 111	37	1	1	4	
NOP	No operations	•	•	•	•	•	•	•	•	00 000 000	00	1	1	4	
HALT	CPU halted	•	•	•	•	•	•	•	•	01 110 110	76	1	1	4	
DI[3]	IFF_1 ← 0 IFF_2 ← 0	•	•	•	•	•	•	•	•	11 110 011	F3	1	1	4	
EI[3]	IFF_1 ← 1 IFF_2 ← 1	•	•	•	•	•	•	•	•	11 111 011	FB	1	1	4	
IM 0[4]	Set interrupt mode 0	•	•	•	•	•	•	•	•	11 101 101 01 000 110	ED 46	2	2	8	
IM 1[4]	Set interrupt mode 1	•	•	•	•	•	•	•	•	11 101 101 01 010 110	ED 56	2	2	8	
IM 2[4]	Set interrupt mode 2	•	•	•	•	•	•	•	•	11 101 101 01 011 110	ED 5E	2	2	8	

Notes: The V symbol in the P/V flag column indicates that the P/V flags contains the overflow of the operation. Similarly the P symbo indicates parity.
[1] F5 and F3 are a copy of bit 5 and 3 of register A
[2] H contains the previous carry state (after instruction $\overline{H}$ ↔ C)
[3] No interrupts are issued directly after a DI or EI.
[4] This instruction has other unofficial opcodes, see Opcodes list.
CY means the carry flip-flop.

Flag Notation: • = flag is not affected, 0 = flag is reset, 1 = flag is set, ↕ = flag is set according to the result of the operation.

4.2.7 Rotate and Shift Group

Mnemonic	Symbolic Operation	Flags S	Z	F5	H	F3	P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
RLCA		•	•	↕	0	↕	•	0	↕	00 000 111	07	1	1	4	
RLA		•	•	↕	0	↕	•	0	↕	00 010 111	17	1	1	4	
RRCA		•	•	↕	0	↕	•	0	↕	00 001 111	0F	1	1	4	
RRA		•	•	↕	0	↕	•	0	↕	00 011 111	1F	1	1	4	
RLC r		↕	↕	↕	0	↕	P	0	↕	11 001 011 00 000 r	CB	2	2	8	r Reg. 000 B
RLC (HL)		↕	↕	↕	0	↕	P	0	↕	11 001 011 00 000 110	CB	2	4	15	001 C 010 D
RLC (IX + d)		↕	↕	↕	0	↕	P	0	↕	11 011 101 11 001 011 ← d → 00 000 110	DD CB	4	6	23	011 E 100 H 101 L 111 A
RLC (IY + d)		↕	↕	↕	0	↕	P	0	↕	11 111 101 11 001 011 ← d → 00 000 110	FD CB	4	6	23	
LD r,RLC (IX + d)*	r ← (IX + d) RLC r (IX + d) ← r	↕	↕	↕	0	↕	P	0	↕	11 011 101 11 001 011 ← d → 00 000 r	DD CB	4	6	23	
LD r,RLC (IY + d)*	r ← (IY + d) RLC r (IY + d) ← r	↕	↕	↕	0	↕	P	0	↕	11 111 101 11 001 011 ← d → 00 000 r	FD CB	4	6	23	
RL m		↕	↕	↕	0	↕	P	0	↕	010					Instruction format
RRC m		↕	↕	↕	0	↕	P	0	↕	001					and states are the
RR m		↕	↕	↕	0	↕	P	0	↕	011					same as RLC.
SLA m		↕	↕	↕	0	↕	P	0	↕	100					Replace 000 with
SLL m*		↕	↕	↕	0	↕	P	0	↕	110					new number.
SRA m		↕	↕	↕	0	↕	P	0	↕	101					
SRL m		↕	↕	↕	0	↕	P	0	↕	111					
RLD		↕	↕	↕	0	↕	P	0	•	11 101 101 01 101 111	ED 6F	2	5	18	
RRD		↕	↕	↕	0	↕	P	0	•	11 101 101 01 100 111	ED 67	2	5	18	

Notes: The P symbol in the P/V flag column indicates that the P/V flags contains the parity of the result.
r means any of the registers A, B, C, D, E, H, L.
* means unofficial instruction.
CY means the carry flip-flop.

Flag Notation: • = flag is not affected, 0 = flag is reset, 1 = flag is set, ↕ = flag is set according to the result of the operation.

4.2.8 Bit Manipulation Group

Mnemonic	Symbolic Operation	S	Z	F5	H	F3	P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
BIT b, r	Z ← $\overline{r_b}$	↕[1]	↕	↕[2]	1	↕[3]	↕[4]	0	•	11 001 011 01 b r	CB	2	2	8	r Reg. 000 B
BIT b, (HL)	Z ← $\overline{(HL)_b}$	↕[1]	↕	↕[2]	1	↕[3]	↕[4]	0	•	11 001 011 01 b 110	CB	2	3	12	001 C 010 D
BIT b, (IX + d)[5]	Z ← $\overline{(IX + d)_b}$	↕[1]	↕	↕[2]	1	↕[3]	↕[4]	0	•	11 011 101 11 001 011 ← d → 01 b 110	DD CB	4	5	20	011 E 100 H 101 L 111 A
BIT b, (IY + d)[5]	Z ← $\overline{(IY + d)_b}$	↕[1]	↕	↕[2]	1	↕[3]	↕[4]	0	•	11 111 101 11 001 011 ← d → 01 b 110	FD CB	4	5	20	
SET b, r	r_b ← 1	•	•	•	•	•	•	•	•	11 001 011 $\underline{11}$ b r	CB	2	2	8	b Bit. 000 0 001 1
SET b, (HL)	$(HL)_b$ ← 1	•	•	•	•	•	•	•	•	11 001 011 $\underline{11}$ b 110	CB	2	4	15	010 2 011 3
SET b, (IX + d)	$(IX + d)_b$ ← 1	•	•	•	•	•	•	•	•	11 011 101 11 001 011 ← d → $\underline{11}$ b 110	DD CB	4	6	23	100 4 101 5 110 6 111 7
SET b, (IY + d)	$(IY + d)_b$ ← 1	•	•	•	•	•	•	•	•	11 111 101 11 001 011 ← d → $\underline{11}$ b 110	FD CB	4	6	23	
LD r,SET b, (IX + d)*	r ← (IX + d) r_b ← 1 (IX + d) ← r	•	•	•	•	•	•	•	•	11 011 101 11 001 011 ← d → $\underline{11}$ b r	DD CB	4	6	23	
LD r,SET b, (IY + d)*	r ← (IY + d) r_b ← 1 (IY + d) ← r	•	•	•	•	•	•	•	•	11 111 101 11 001 011 ← d → $\underline{11}$ b r	FD CB	4	6	23	
RES b, m	m_b ← 0 m ≡ r, (HL), (IX+d), (IY+d)	•	•	•	•	•	•	•	•	$\underline{10}$					To form new opcode replace $\underline{11}$ of SET b, s with $\underline{10}$. Flags and states are the same.

Notes: The notation m_b indicates bit b (0 to 7) of location m.
BIT instructions are performed by an bitwise AND.
[1] S is set if b = 7 and Z = 0
[2] F5 is set if b = 5 and Z = 0
[3] F3 is set if b = 3 and Z = 0
[4] P/V is set like the Z flag
[5] This instruction has other unofficial opcodes
* means unofficial instruction.

Flag Notation: • = flag is not affected, 0 = flag is reset, 1 = flag is set, ↕ = flag is set according to the result of the operation.

4.2.9 Input and Output Groups

Mnemonic	Symbolic Operation	S	Z	F5	H	F3	P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
IN A, (n)	A ← (n)	•	•	•	•	•	•	•	•	11 011 011 ← n →	DB	2	3	11	r Reg. 000 B
IN r, (C)	r ← (C)	↕	↕	↕	0	↕	P	0	•	11 101 101 01 r 000	ED	2	3	12	001 C 010 D
IN (C)* or IN F, (C)*	Just affects flags, value is lost.	↕	↕	↕	0	↕	P	0	•	11 101 101 01 110 000	ED 70	2	3	12	011 E 100 H
INI	(HL) ← (C) HL ← HL + 1 B ← B - 1	↕[1]	↕[1]	↕[1]	↕[3]	↕[1]	X	↕[2]	↕[3]	11 101 101 10 100 010	ED A2	2	4	16	101 L 111 A
INIR	(HL) ← (C) HL ← HL + 1 B ← B - 1 Repeat until B = 0	0	1	0	↕[3]	0	X	↕[2]	↕[3]	11 101 101 10 110 010	ED B2	2 2	5 4	21 16	if B ≠ 0 if B = 0
IND	(HL) ← (C) HL ← HL - 1 B ← B - 1	↕[1]	↕[1]	↕[1]	↕[4]	↕[1]	X	↕[2]	↕[4]	11 101 101 10 101 010	ED AA	2	4	16	
INDR	(HL) ← (C) HL ← HL - 1 B ← B - 1 Repeat until B = 0	0	1	0	↕[4]	0	X	↕[2]	↕[4]	11 101 101 10 111 010	ED BA	2 2	5 4	21 16	if B ≠ 0 if B = 0
OUT (n), A	(n) ← A	•	•	•	•	•	•	•	•	11 010 011 ← n →	D3	2	3	11	
OUT (C), r	(C) ← r	•	•	•	•	•	•	•	•	11 101 101 01 r 001	ED	2	3	12	
OUT (C), 0*	(C) ← 0	•	•	•	•	•	•	•	•	11 101 101 01 110 001	ED 71	2	3	12	
OUTI	(C) ← (HL) HL ← HL + 1 B ← B - 1	↕[1]	↕[1]	↕[1]	X	↕[1]	X	X	X	11 101 101 10 100 011	ED A3	2	4	16	
OTIR	(C) ← (HL) HL ← HL + 1 B ← B - 1 Repeat until B = 0	0	1	0	X	0	X	X	X	11 101 101 10 110 011	ED B3	2 2	5 4	21 16	if B ≠ 0 if B = 0
OUTD	(C) ← (HL) HL ← HL - 1 B ← B - 1	↕[1]	↕[1]	↕[1]	X	↕[1]	X	X	X	11 101 101 10 101 011	ED AB	2	4	16	
OTDR	(C) ← (HL) HL ← HL - 1 B ← B - 1 Repeat until B = 0	0	1	0	X	0	X	X	X	11 101 101 10 111 011	ED BB	2 2	5 4	21 16	if B ≠ 0 if B = 0

Notes: The V symbol in the P/V flag column indicates that the P/V flags contains the overflow of the operation. Similarly the P symbol indicates parity.
r means any of the registers A, B, C, D, E, H, L.
[1] flag is affected by the result of B ← B - 1 as in DEC B.
[2] N is a copy bit 7 of the last value from the input (C).
[3] this flag contains the carry of (((C + 1) AND 255) + (C))
[4] this flag contains the carry of (((C - 1) AND 255) + (C))
* means unofficial instruction.

Flag Notation: • = flag is not affected, 0 = flag is reset, 1 = flag is set, X = flag is unknown,
↕ = flag is set according to the result of the operation.

4.2.10 Jump Group

Mnemonic	Symbolic Operation	S	Z	F5	H	F3	P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
JP nn	PC ← nn	•	•	•	•	•	•	•	•	11 000 011	C3	3	3	10	
										← n →					
										← n →					
JP cc, nn	if cc is true,	•	•	•	•	•	•	•	•	11 ccc 010		3	3	10	<u>ccc Condition</u>
	PC ← nn									← n →					000 NZ
										← n →					001 Z
															010 NC
															011 C
															100 PO
															101 PE
															110 P
JR e	PC ← PC + e	•	•	•	•	•	•	•	•	00 011 000	18	2	3	12	111 M
										← e - 2 →					
JR ss, e	if ss is true	•	•	•	•	•	•	•	•	00 ss 000		2	3	12	if ss is true
	PC ← PC + e									← e - 2 →		2	2	7	if ss is false
JP HL	PC ← HL	•	•	•	•	•	•	•	•	11 101 001	E9	1	1	4	
JP IX	PC ← IX	•	•	•	•	•	•	•	•	11 011 101	DD	2	2	8	<u>ss Condition</u>
										11 101 001	E9				111 C
															110 NC
JP IY	PC ← IY	•	•	•	•	•	•	•	•	11 111 101	FD	2	2	8	101 Z
										11 101 001	E9				100 NZ
DJNZ e	B ← B - 1	•	•	•	•	•	•	•	•	00 010 000	10	2	2	8	if B = 0
	if B ≠ 0									← e - 2 →		2	3	13	if B ≠ 0
	PC ← PC + e														

Notes: e is a signed two-complement number in the range <-126, 129>
e - 2 in the opcode provides an effective number of PC + e as PC incremented by 2 prior to the addition of e.

Flag Notation: • = flag is not affected, 0 = flag is reset, 1 = flag is set, ↕ = flag is set according to the result of the operation.

4.2.11 Call and Return Group

	Symbolic				Flags					Opcode		No.of	No.of M	No.of T	
Mnemonic	Operation	S	Z	F5	H	F3	P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments
CALL nn	SP ← SP - 1	•	•	•	•	•	•	•	•	11 001 101	CD	3	5	17	
	(SP) ← PC_H									← n →					
	SP ← SP - 1									← n →					
	(SP) ← PC_L														
	PC ← nn														
CALL cc, nn	if cc is true,	•	•	•	•	•	•	•	•	11 ccc 100		3	3	10	if cc is false
	SP ← SP - 1									← n →		3	5	17	if cc is true
	(SP) ← PC_H									← n →					
	SP ← SP - 1														
	(SP) ← PC_L														
	PC ← nn														
RET	PC_L ← (SP)	•	•	•	•	•	•	•	•	11 001 001	C9	1	3	10	
	SP ← SP + 1														
	PC_H ← (SP)														
	SP ← SP + 1														
RET cc	if cc is true,	•	•	•	•	•	•	•	•	11 ccc 000		1	1	5	if cc is false
	PC_L ← (SP)											1	3	11	if cc is true
	SP ← SP + 1														
	PC_H ← (SP)														
	SP ← SP + 1														
RETI[2]	PC_L ← (SP)	•	•	•	•	•	•	•	•	11 101 101	ED	2	4	14	cc Condition
	SP ← SP + 1									01 001 101	4D				000 NZ
	PC_H ← (SP)														001 Z
	SP ← SP + 1														010 NC
															011 C
RETN[1,2]	PC_L ← (SP)	•	•	•	•	•	•	•	•	11 101 101	ED	2	4	14	100 PO
	SP ← SP + 1									01 000 101	45				101 PE
	PC_H ← (SP)														110 P
	SP ← SP + 1														111 M
	IFF_1 ← IFF_2														
RST p	SP ← SP - 1	•	•	•	•	•	•	•	•	11 t 111		1	3	11	t p
	(SP) ← PC_H														000 0h
	SP ← SP - 1														001 8h
	(SP) ← PC_L														010 10h
	PC ← p														011 18h
															100 20h
															101 28h
															110 30h
															111 38h

Notes: [1] This instruction has other unofficial opcodes, see Opcode list.
[2] Instruction also IFF_1 ← IFF_2

Flag Notation: • = flag is not affected, 0 = flag is reset, 1 = flag is set, ↕ = flag is set according to the result of the operation.

4.3 Glossary

Applet	- An internet application that runs inside an internet browser.
Bank	- A Computer science term used to describe a specific chunk of Random Access Memory (See Random Access Memory).
BIOS	- Basic Input and Output System. A set of programs, addresses or routines inside RAM (See Random Access Memory) that provide certiain functionality for the computer system.
Bit	- The smallest value used to represent computer data or memory in a base 2 binary numbering system having a value of 1 or 0.
Buad Rate	- A term used to describe the ability of two devices ports (See Port) to establish a communication between them at a certain speed of data transfer.
Buffer	- A permanent or temporary area of storage used to hold data.
Byte	- A standard unit of measurement for computer data or RAM (See Random Access Memory)
CPU	- The Central Processing Unit.
DOS	- Disk Operating System.
I/O	- Input and Output.
Interrupt	- A term used to describe the need for a device or software program that must send a message to the Processor (See CPU) in order to gain its attention for useage.
Java	- A programming language with internet and platform independent capibility.
Memory	- Term used to describe an area of storage in a computer system. (See Random Access Memory, Bank, Buffer)
Memory Mapped	- A term used to describe how a computer system connects it I/O (See I/O) to RAM (See Random Access Memory).
OP Code	- The basic unit of instruction in a computer system. This is what is executed when a computer program is running.
Operating system	- The software program that manages low level hardwareand software management inside a computer system.
Parallel	- Data transmission that occurs in a side by side manor using multiple data lines to transmit data across a specific line.
Port	- A term used to describe the means for I/O (See I/O) internally and externally in a computer system.
RAM	- See Random Access Memory.

Random Access Memory	- The second fastest form of storage used inside a computer system commonly used for application execution and data storage.
Register	- The fastest form a storage inside a computer system usually constrained to a finite size depending on a particular system.
ROM	- Read Only Memory. Usually contains useful programs or data for Operating System (See Operating System), hardware and program useage.
Serial	- A term used to describe inline communication or data that is sent one after another either interanally or externally.
Virtual Machine	- A term used to describe a software or hardware program that emulates a given environment in which its executing applications are thought to be running.