Abstract:

This project deals with the simulation of 16-bit general-purpose microprocessor using VHDL. The microprocessor developed, is mainly for general computational purposes.

The basic structure and logical functionality of the processor is based on the Intel 8085 microprocessor. The top-level architecture is divided into six functional blocks: precoder block, decoder block, ALU (Arithmetic-Logical Unit), register block, interrupt block and finally output block.

We have satisfactorily coded and simulated the entire microprocessor. Future work would involve expanding the opcode list to include any extra functions that we have overlooked due to the time constraints of the project.

Contents:

Introduction:

Microprocessors and microcontrollers form the core of all computers and electronic gadgets in the modern world, so much so that the term electronics is slowly but steadily becoming synonymous with both of them.

Microprocessor is the most important device studied by any electronics student in his undergraduate curriculum, which is partly the reason that we chose to do this particular project.

In this project, we have successfully coded and simulated a 16-bit CISC general-purpose microprocessor using VHDL. The overall structure of our microprocessor is based on the Intel 8085 general-purpose microprocessor.

Our microprocessor is divided into six functional blocks based on the functions they perform. They are: precoder block, decoder block, ALU (Arithmetic-Logical Unit), register block, interrupt block and finally output block. The codes and simulation results along with their explanations are given later in the report.

The clock frequency of the clock for our microprocessor is set at 100MHz, i.e. the clock period is 10ns. This frequency we have determined purely on the basis of simulation results.

We have undertaken this project mainly on a study basis and we have tried to make our microprocessor as efficient and close to a standard general-purpose microprocessor as possible within the time allotted to us.

Introduction to VHDL:

a. What is VHDL?

VHDL is acronym for VHSIC (Very High Speed Integrated Circuit) Hardware Description Language. It is an industry standard hardware description language for electronic design automation used to describe both digital and mixed-signal systems such as FPGA (Field Programmable Gate Array) and Integrated Circuits.

It can be used to model a digital system at any level of abstraction from the algorithmic to the gate level.

b. Why use VHDL?

Although VHDL is considered tougher to understand than other hardware description languages, we have chosen to implement our project in VHDL as it offers the following advantages:

- VHDL is an IEEE and ANSI standard. This increases portability between tools, companies and products. It means that VHDL hardware designs and test benches are portable across the multiple platforms of various design tool vendors available in the market.
- VHDL was designed to be technology independent, implying that if a system was described today using VHDL then it can be implemented using any previous or future technology if available.
- Behavioral simulation can reduce design times by allowing design problems to be detected early on, thereby avoiding the need to rework designs at the gate level.
- Behavioral simulation also helps in exploring various alternative architectures, resulting in better designs.
- It supports a wide range of abstraction level ranging from behavioral description to very precise gate level descriptions.
- VHDL allows the description of a concurrent system. VHDL is a dataflow language, unlike procedural computing languages such as BASIC, C, and assembly code, which all run sequentially, one instruction at a time.
- It supports both synchronous and asynchronous timing models.
- The language supports three basic different description styles: structural, data flow and behavioral. A design may be described in any combination of these three descriptive styles.
- The language has element that make large-scale modeling easier, for example component, functions, procedure, and packages.

c. Overview of VHDL:

It was originally developed in the 1980s by the United States Department of Defense. VHDL was developed as a tool for designers to synthesize and simulate circuits. However, though

VHDL is fully simulatable, it is not fully synthesizable. It means that certain codes used in VHDL cannot be synthesized at the hardware level.

Like its predecessor Ada, VHDL is a strongly typed language and is not case sensitive. The models written in this language can be verified using a VHDL simulator. VHDL is often considered difficult to understand due to its extensive range of modeling capabilities. However, it is possible to quickly understand the essential features of the language without learning its more complex features. These subsets of features are usually sufficient to model most applications.

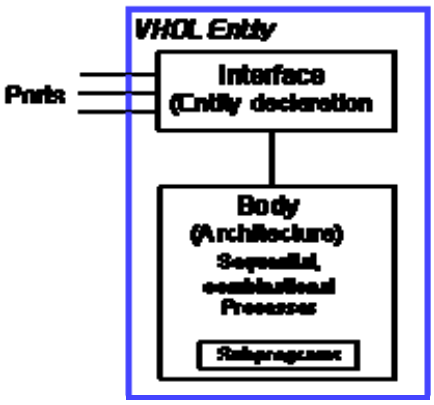
A digital system in VHDL consists of a design entity that can contain other entities, which are then considered as components of the top-level entity. Each entity is modeled by an entity declaration and an architecture body.

One can consider the entity declaration as the interface to the outside world that defines the input and output signals, while the architecture body contains the description of the entity and is composed of interconnected entities, processes and components, all operating concurrently, as schematically shown in the figure below.

This is, contrary to regular computer programs where the statements are executed sequentially. This is the reason that VHDL is referred to as a code, rather than a program. In fact, only those statements present within a PROCESS, FUNCTION or a PROCEDURE are executed sequentially.

In a typical design there will be many such entities connected together to perform the desired function.

VHDL uses reserved keywords that cannot be used as signal names or identifiers. Keywords and user-defined identifiers are case insensitive.

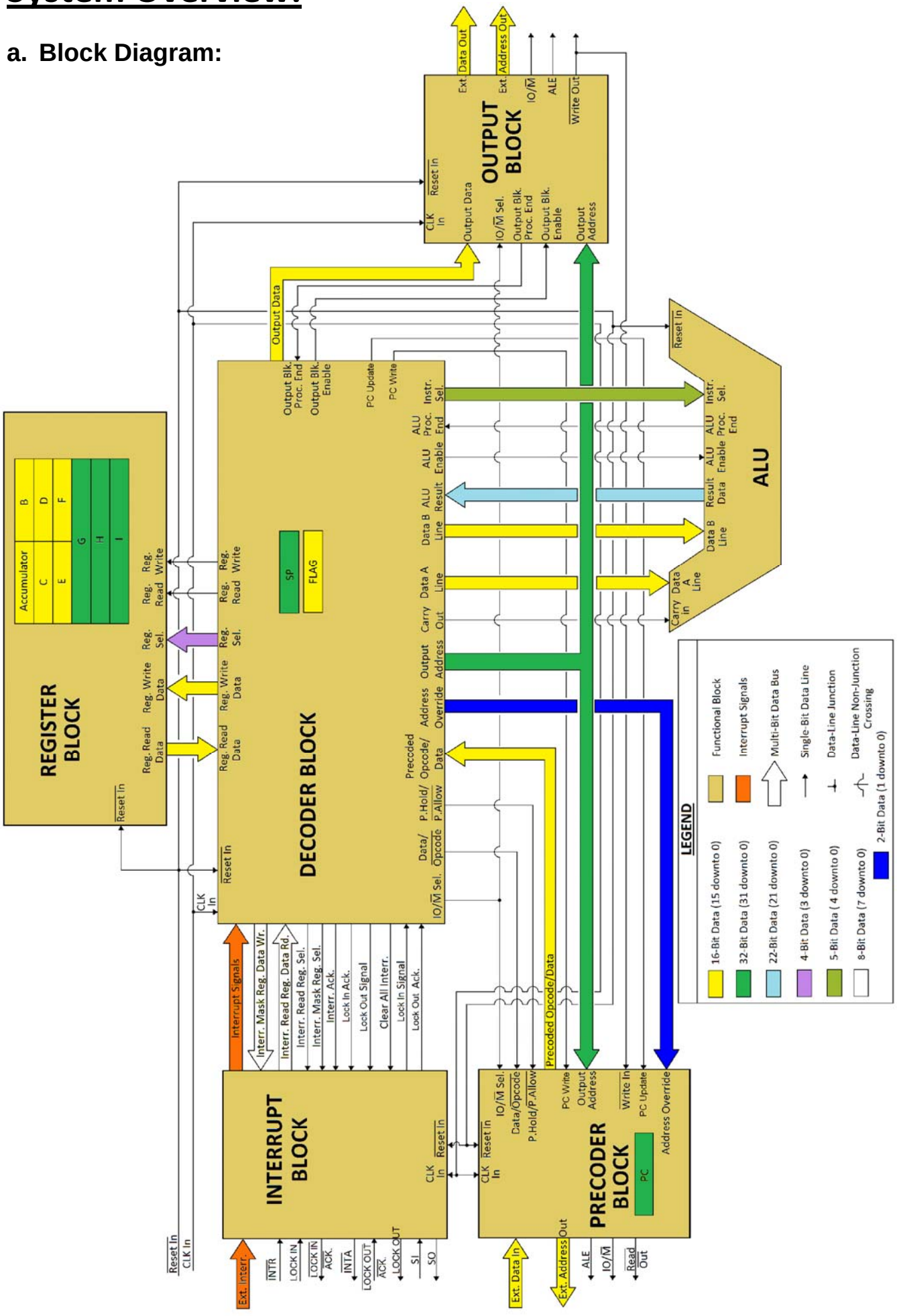


Lines with comments start with two adjacent hyphens (--) and will be ignored by the compiler. VHDL also ignores line breaks and extra spaces.

VHDL naturally leads to a top-down design methodology, where the system is first specified at high level and is tested using a simulator. After successful simulation, the system is then gradually refined, eventually leading to a structural description, closely related to the actual hardware implementation.

System Overview:

a. Block Diagram:



b. General Description:

The microprocessor is divided into seven functional blocks, each of which is concerned with particular functions as mentioned below:

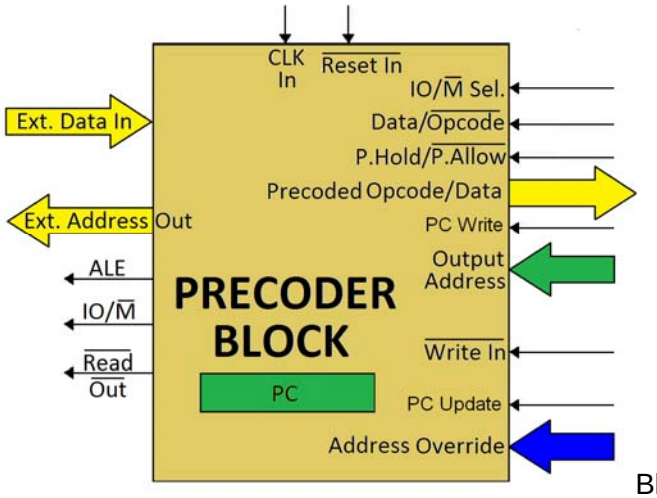
i. Precoder Block: The precoder block mainly concerned with the pipelining and precoding of the instructions. From the block diagram of the precoder block, we can see that it has the following input and output ports:

- *Reset In*: The external reset signal is connected to this input signal port. This is an active LOW port, which on being activated, resets the precoder block.

- *CLK In*: This is the input port for the external clock signal, used to synchronize the microprocessor with the external peripherals.

- *IO/M Sel.*: This input port is used by the IO/M signal from the decoder. This signal tells the precoder if the next data to be fetched is from an IO device or from memory.

- *Data/Opcode*: This input signal tells the precoder if the next data expected by the *Decoder Block* is an opcode or a data.



Block Diag. of Precoder Block

- *Precoded Opcode/Data*: This is a 16-bit bus that sends the precoded opcode or raw data to the *Decoder Block* as is required.

- *Output Address*: This is a 16-bit bus, which contains the value of the address to which the microprocessor execution needs to be transferred to.

- *Write In*: This input port is linked to the *Write Out* signal of the *Output Block*. Write operation has a precedence over read operation, and hence, if this signal is active, then the *Precoder Block* temporarily ceases to read external data until the signal is deactivated. This signal is active LOW.

- *Address Override:* This is a 2-bit bus that updates the value of the the internal address counter of the precoder block or reads the value of PC as per the truth table as follows:

Address Override value	Operation
00	No operation
01	Updates the internal address counter with the value of the *Output Address* bus
10	Reads the value of PC in the subsequent clocks
11	Updates the internal address counter with the value of PC

- *Read Out*: This is an output port used to inform the external device that the microprocessor is ready to read the next data/instruction. This signal is active LOW.

- *IO/M*: This is an output port. It is different from the previous *IO/M Sel*. port. This output signal is used to inform the peripherals whether the next data is to be read from a memory or an I/O.

- *ALE*: This is the *Address Latch Enable* port. It is an output port, which is used to demultiplex the address from the data.

- *Ext. Address Out*: This is a 16-bit data output bus, used as external address for the reading of the data.

- *Ext. Data Out*: This is a 16-bit data input port, used by the precoder to read the incoming data/instruction from the memory (or I/O) location addressed by the *Ext. Address Out* port. This bus is demultiplexed using the *ALE* into 16 higher order bits of address + 16 data bits.

- *PC Update*: This is a single bit line, which on being activated, increments the value of the program counter (PC) by unity.

- PC Write: This is a single bit line, which on being activated, updates the value of the program counter (PC) with the value of the *Output Address* bus.

ii. Interrupt Block: The purpose of the *Interrupt Block* is to monitor the various interrupt signals and send appropriate signals to the *Decoder Block*. It contains the *Interrupt Mask Register* and the *Interrupt Read Register*, which can be written to and read by the decoder respectively, to mask the interrupt signals when needed. The ports of the *Interrupt Block* are:

- *Reset In*: This is an input signal port to which the external reset signal is connected. This is an active LOW port, which on being activated, resets the interrupt block.

- *CLK In*: This is the input port for the external clock signal, used to synchronize the microprocessor with the external peripherals.

- *Ext. Interr.*: These are the various hardware interrupt signals that are to be used to interrupt the microprocessor. Among them, few have higher priorities than others do, while some of them are unmaskable, meaning that the interrupts coming to those ports cannot be masked by the *Interrupt Mask Register*.

- *INTR*: This is the *Interrupt Request* signal that is received from an external device, which wants to interrupt the operation of the microprocessor.

- *LOCK IN*: This is a single bit input signal from an external device, which upon activation indicates that that peripheral wishes to gain control of the external address and data bus lines and is requesting this microprocessor to relinquish its hold over the bus lines.

- *INTA*: This single bit output signal is used microprocessor to acknowledge the *INTR* request.

- *LOCK IN ACK.*: This single bit output signal is used by the microprocessor to acknowledge the LOCK IN request.

- *LOCK OUT*: This single bit output signal upon activation, effectively locks the data and the address bus lines by disabling other devices from accessing the bus lines. It is obvious that this line enters another microprocessor through the *LOCK IN* port.

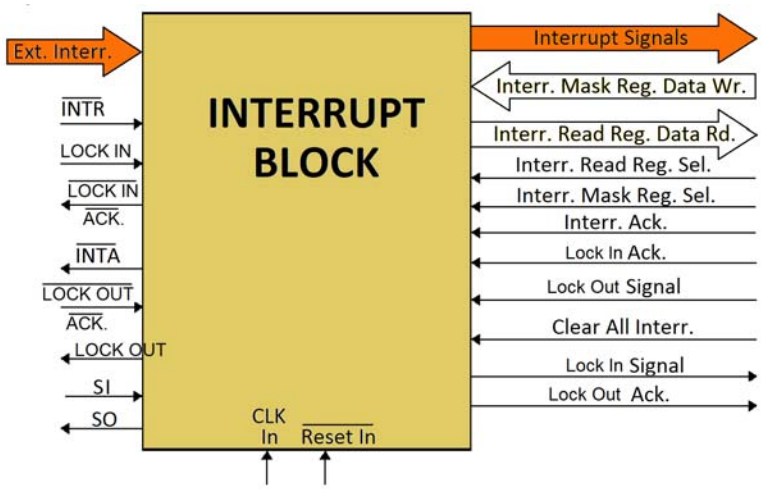
- *LOCK OUT ACK.*: This single bit input signal received by the microprocessor in acknowledgement to its *LOCK OUT* request.

- *SI*: This single bit input port is used for serial input communication with a serial device.

- *SO*: This single bit output port is used for serial output communication with a serial device.

- *Clear All Interr.*: This single bit input signal clears all pending interrupts.

- *Lock out Signal*: This is a single bit input signal from the decoder, the value of which activates (for HIGH) or deactivates (for LOW) the *LOCK OUT* signal.



Block Diag. of Interrupt Block

- *Lock In Ack*.: This is a single bit input signal from the decoder, the value of which is used to set or reset the *LOCK IN ACK.* Signal

- *Lock In Signal*: This is a single bit output signal used to inform the decoder that a *LOCK IN* request has occurred.

- *Lock Out Ack.:* This is a single bit output signal used to inform the decoder that the *LOCK OUT* request has been acknowledged.

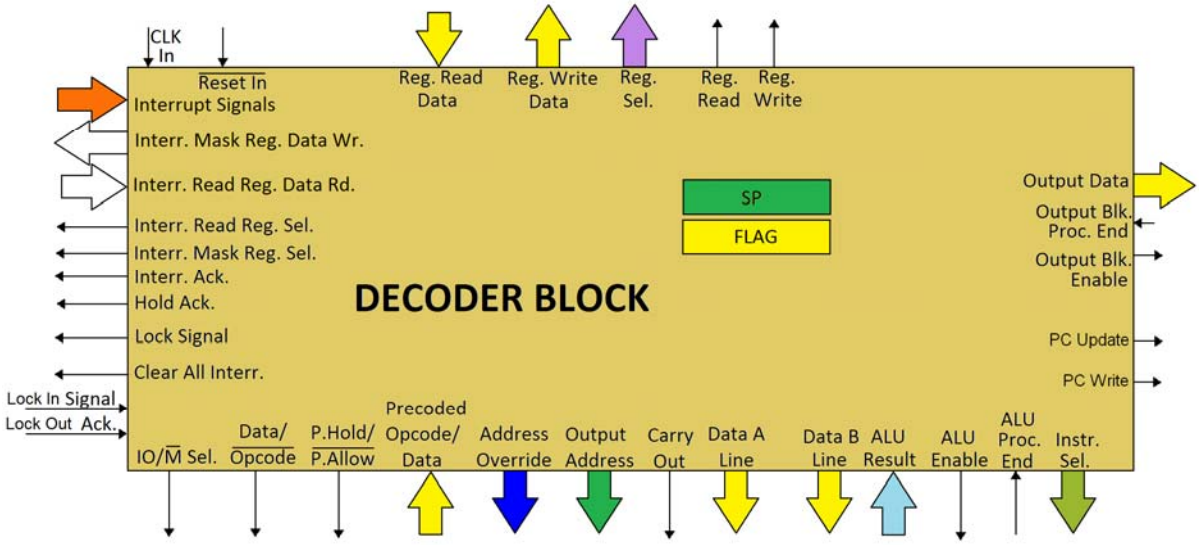
- *Interr*. *Ack*.: This is a single bit input signal from the decoder, the value of which is used to set or reset the *INTA* signal.

- *Interr. Mask Reg. Sel. & Interr. Mask Reg. Data Wr.*: *Interr. Mask Reg.* is a single bit input signal used along with 8-bit *Interr. Mask Reg. Data Wr.* bus line to overwrite the contents of the *Interrupt Mask register* by the *Decoder Block*.

- *Interr. Read Reg. Sel. & Interr. Read Reg. Data Rd.*: *Interr. Read Reg.* is a single bit input signal used along with 8-bit *Interr. Read Reg. Data Rd.* bus line to read the contents of the *Interrupt Read register* by the *Decoder Block*.

- *Interrupt Signals*: These are the various output interrupt signals generated by the *Interrupt block* from the *Ext. Interr.* signals, *INTR* signal, *HOLD* signal and the *Interrupt Mask register* for the *Decoder Block*.

iii. Decoder Block: The *Decoder Block* is the heart of the microprocessor. It is the block that finally decodes the precoded instruction (taken from the *Precoder Block*) and takes the appropriate steps needed to execute the instructions. It also monitors for interrupts and takes the appropriate steps when interrupt occurs. The *Decoder Block* contains the PC (program counter), SP (stack pointer) and Flag register within it in order to speed up the execution of JUMP or other such instructions. The decoder communicates with the other blocks using the following ports:



Block Diag. of Decoder Block

- *Reset In*: This is an input signal port to which the external reset signal is connected. This is an active LOW port, which on being activated, resets the decoder block.

- *CLK In*: This is the input port for the external clock signal, used to synchronize the microprocessor with the external peripherals.

- *IO/M Sel., Data/Opcode, Precoded Opcode/Data, Address Override, PC Update, PC Write & Output Address: Refer to Precoder Block*.

- *Reg. Read Data*: This is a 16-bit input port that is used to read the data from the *Register Block*.

- *Reg. Write Data*: This is a 16-bit output port, which is used to write data to the *Register Block*.

- *Reg. Sel.*: This is an output port, which is used to select the register in the *Register Block*, on which the read/write operation is to be performed.

- *Reg. Read*: This is a single bit output signal port, which is used to inform the *Register Block* that a read operation is to be performed.

- *Reg. Write*: This is a single bit, output signal port, which is used to inform the *Register Block* that a write operation is to be performed.

- *Output Data*: This is a 16-bit output port, which is used to give the *Output Block* the data to be written to the external memory or I/O.

- *Output Blk. Proc. End*: This is a single bit input port from the *Output Block*, the activation of which signifies that the *Output Block* has finished writing the data.
- *Output Blk. Enable*: This is a single bit output port to the *Output Block*, which upon activation gives the *Output Block* clearance to start its execution.

- *Instr. Sel*.: This is a 5-bit output port, which tells the *ALU* which operation is to be performed by it.

- *ALU Enable*: It is a single bit output port, which gives the *ALU* clearance to begin its execution.

- *Data A & B Line*: They are 16-bit output ports that carry the data to the *ALU* to be operated on.

- *Carry Out*: It is a single bit output port, which carries the value of the carry to the *ALU* as per the requirement of the operation.

- *ALU Result*: It is a 22-bit input port, which is used to read the result of the *ALU* operation and the carry and overflow flag values from the *ALU* to the *Decoder Block*.

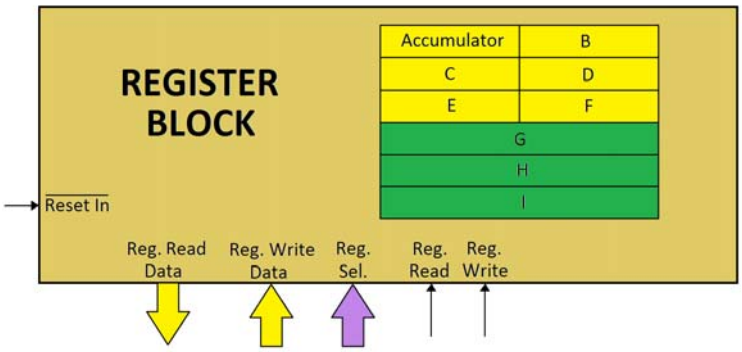
- *ALU Proc. End*: This is a single bit input port from the *ALU*, the activation of which signifies that the *ALU* has finished operating on the data.

- *Clear All Interr*., *Lock Out Signal, Lock In Ack., Lock Out Ack. Lock In Signal, Interr. Ack. Interr. Mask Reg. Sel*. *& Interr. Mask Reg. Data Wr*., *Interr. Read Reg. Sel*. *& Interr. Read Reg. Data Rd*., *Interrupt Signals*: *Refer to Interrupt Block.*

iv. Register Block: The *Register Block* contains the various internal registers of the microprocessor (except a few), used to store the various intermediate values generated by the operations of the processor and are also used by programmers for data manipulation and logic implementation. It consists of the 16-bit accumulator, five 16-bit registers (B,C,D,E,and F) and three 32-bit register pairs (G,H,L). It consists of the following ports:

- *Reset In*: This is an input signal port to which the external reset signal is connected. This is an active LOW port, which on being activated, resets the register block.

- *Reg. Read Data, Reg. Write Data, Reg. Sel., Reg. Read & Reg. Write*: *Refer to the Decoder Block*.

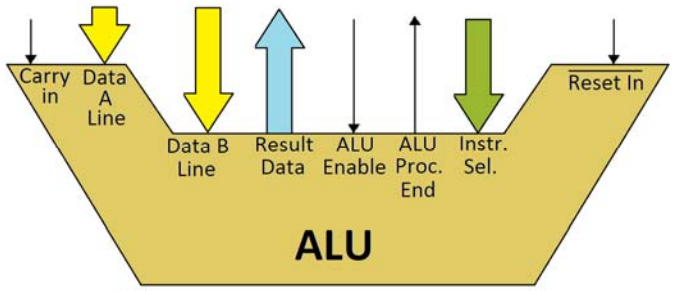


Block Diag. of Register Block

v. ALU: The purpose of the *ALU* is to perform the arithmetic and logical operations on the data given to it. It performs various arithmetic operations like ADD, SUB, etc. as well as various logical operations like SHIFT etc. It has the following ports:

- *Reset In*: This is an input signal port to which the external reset signal is connected. This is an active LOW port, which on being activated, resets the ALU block.

- *Carry In, Data A & B Line, Result Data, ALU Enable, ALU Proc. End & Instr. Sel.: Refer to Decoder Block.*



Block Diag. of ALU

vi. Output Block: The purpose of the *Output Block* is to provide a means for the microprocessor to write the required data (or result from ALU) in an external memory (or I/O) space. It has the following ports:

- *Reset In*: This is an input signal port to which the external reset signal is connected. This is an active LOW port , which on being activated, resets the output block.

- *CLK In*: This is the input port for the external clock signal, used to synchronize the microprocessor with the external peripherals.

- *IO/M Sel., Write In: Refer to Precoder Block.*

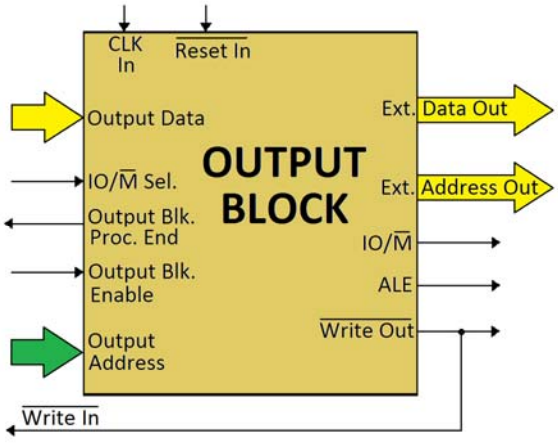
- *Output Data, Output Blk. Proc. End, Output Blk. Enable, Output Address, LSB Sel.: Refer to Decoder Block.*

- *IO/M*: This is an output port. It is different from the previous *IO/M Sel*. port. This output signal is used to inform the peripherals whether the next data is to be written to a memory or an I/O.

- *ALE*: This is the *Address Latch Enable* port. It is an output port, which is used to demultiplex the address from the data.

- *Ext. Address Out*: This is a 16-bit data output bus, used as external address for the writing of the data.
- *Ext. Data Out*: This is a 16-bit data output port, used by the *Output Block* to write the data (or result) to the memory (or I/O) location addressed by the *Ext. Address Out* port. This bus is demultiplexed using the *ALE* into 16 higher order bits of address + 16 data bits.

- *Write Out*: This is a single bit output port used to inform the external device that the microprocessor is ready to write a data. This is an active LOW signal.



Block Diag. of Output Block

c. Sample Instruction Representation:

The instructions to be used in the microprocessor will be of the following format:

Type 1: <Mnemonic> <Primary Register>, <Secondary Register>
For e.g.: MOV A,B; MOV A,M

This will be a 16-bit opcode. Here, the operation will be performed on the value contained by one or both the registers and the final value will be stored in the primary register. For e.g., in the MOV A,B operation, the contents of B register will be moved to the A register.

Type 2: <Mnemonic> <Primary Register>
For e.g.: ADD B; SUB C

This will be a 16-bit opcode. Here, the operation will be done between the value of the mentioned register and accumulator. The result of the operation will be stored in the accumulator. This type will mainly contain ALU operation related opcodes, among others.

Type 3: <Mnemonic> <32-bit address>
For e.g.: LDA XXXX1078 H

This opcode will require 3*16=48 bits to be stored. This type will load accumulator or other register (as mentioned in the opcode) with the data located at the 32-bit address mentioned.

Type 4: <Mnemonic> <16-bit/32-bit Data>
For e.g.: MVI A 2030 H; LXI H XXXX5060 H

This opcode will require 2*16=32 bits to 3*16=48 bits to be stored. This will load the register (or register pair) or accumulator with the adjacent data (or data pair), or will perform some operation on the adjacent data and the value of the accumulator or other register (according to the opcode) and store the data accordingly.

Type 5: <Mnemonic><16-bit data><32-bit address>
For e.g.: JE 1254 H,XXXX1290 H

This will require 4*16=64 bits to be stored. This is mainly those type of opcodes that will perform some logical operation on the data provided and based on the result, it will execute a branching operation like JUMP or CALL and if the condition, mentioned in the mnemonic is valid, then program execution will be branched to the 32-bit address provided in the opcode.

Type 6: <Mnemonic>
For e.g.: EI; DI

This will be a 16-bit opcode. They will be simple instructions like enable or disable interrupt, or reset the carry bit, or other such instructions.

For detailed opcode table please refer to Appendix - A.

Salient features of the Microprocessor architecture:

One of the main overall improvements that we have implemented over the internal architecture of the Intel 8085 microprocessor (that we have based our microprocessor on) is that we have changed the internal architecture design style from von Neumann to Harvard style.

In doing so, we have essentially eliminated the von Neumann bottlenecks that occur in von Neumann style of architecture design (which is followed by the Intel 8085). These bottlenecks arise because all the blocks in 8085 communicate through a common bus. Therefore, when one block accesses the bus, the others have to wait for their turn to come.

We have eliminated that problem by using Harvard style of design architecture, where every block communicates with the central decoder using their own independent buses, thereby eliminating bottlenecks and improving efficiency and throughput of the device. The other features and changes that we have implemented are block wise listed as follows:

Precoder Block:

- The precoder block is an improvement over the architecture of the 8085 microprocessor. The main function of this block is to pipeline the flow of instructions and data into the microprocessor and to 'precode' the opcode entering the microprocessor.

- It does it in two parts. The first part deals with the pipelining of the data flow.

- This is achieved by using a circular queue in temporary storing of the data in this block. The queue has two pointer registers associated with it, the input pointer and the output pointer.

- The input pointer will contain the address of the 16-bit register to which the data from the external source is to be written to, and the output pointer will contain the address of the 16-bit register from which the data will be sent from the precoder to the decoder. The precoder will stop reading only when the entire queue is full.

- In case of branching instructions, there is provision to change the value of the address counter, which keeps track of the current output address. This is done by giving the desired output address in the *Output Address* port and in the *Address Override* Signal by the decoder.

- There is also a provision to change the value of the program counter (PC), by using the *PC Write Signal*. After the decoding and execution of each instruction, the value at PC is incremented by using the *PC Update Signal*, both originate from the decoder and terminate at the precoder.

- The *External Data Signal* is a multiplexed data/address signal, which is demultiplexed into 16 higher order address bits and 16 data bits by the use of the ALE signal.

- In the second part, the precoder precodes the instructions so that it is easier for the decoder to decode the instruction.

- Depending on the input from the decoder in the *Data/Opcode* line, the precoder then transmits the precoded opcode to the decoder or it transmits the raw data to the decoder block.

- The decoder, by using the *P.Hold/P.Allow* signal, indicates if it is busy performing the operations of the previous instruction, or if it is ready to accept the next data/instruction from the precoder.

- There is also a *Write In* signal to the precoder block. Write operation has a preference over read operation, and so when this signal is activated by the output block, the precoder surrenders the control of the bus to the output block and waits for the signal to deactivate.

Interrupt Block:

- The interrupt block mainly deals with, as the name suggests, interrupts of the microprocessor. It is responsible for continuous monitoring of external interrupts by the peripherals to the microprocessor and alerts whenever one occurs.

- It is also used by the microprocessor to issue a LOCK OUT command or acknowledge another device's LOCK IN request with the help of their respective pins.

- The interrupt block contains two registers: the Interrupt Mask register and the Interrupt Read register. The decoder uses the Interrupt Mask register to mask (or block) maskable interrupts and it reads the pending interrupts using Interrupt Read register.

- Apart from this, these registers also help in serial communication of the microprocessor with serial devices using the *SI* and the *SO* ports, using the commands RDI and STI (input and output respectively).

- The microprocessor has four hardware interrupts namely TRAP, RST 7.5, RST 6.5 and RST 5.5, out of which TRAP is the only non-maskable interrupt. The others can be masked using the command STI and the masks along with the interrupt status can be read using the command RDI. The software interrupts are masked by the EI/DI flag.

- The following table gives us the software and hardware interrupts along with their corresponding vectored addresses in increasing order of priority:

Interrupt	Type	Vector Address
RST 0	Software	(00000000) H
RST 1	Software	(00000010) H
RST 2	Software	(00000020) H
RST 3	Software	(00000030) H

Interrupt	Type	Vector Address
RST 4	Software	(00000040) H
RST 5	Software	(00000050) H
RST 5.5	Hardware	(00000058) H
RST 6	Software	(00000060) H
RST 6.5	Hardware	(00000068) H
RST 7	Software	(00000070) H
RST 7.5	Hardware	(00000078) H
RST 8	Software	(00000080) H
RST 9	Software	(00000090) H
RST 10	Software	(000000A0) H
RST 11	Software	(000000B0) H
RST 12	Software	(000000C0) H
RST 13	Software	(000000D0) H
RST 14	Software	(000000E0) H
RST 15	Software	(000000F0) H
TRAP	Hardware	(00000048) H

- The format of the STI and RDI commands is as follows:

STI:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X	SOD	SDE	X	X	MSE	M7.5	M6.5	M5.5

SOD – Serial Output Data
SDE – Serial Data Latch, when ‘1’, it latches the SOD value to the Serial Output port.
MSE – Mask Set Enable, if ‘1’, the mask is set, else if ‘0’, bits 2-0 are ignored.
M7.5-5.5 – Masks the corresponding RST interrupts if set to ‘1’ else the interrupts are available.
X – Don’t Care.

RDI:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X	SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5

SID – Serial Input Data
I7.5-5.5 – These are the status of the corresponding RST interrupts.
IE – It denotes the status of the EI/DI flag.
M7.5-5.5 – These are the status of the masks for the corresponding RST interrupts.
X – Don't Care

Decoder Block:

- The decoder block is the heart of the microprocessor. It is responsible for the proper decoding of the precoded opcode. It is also responsible for taking the appropriate steps needed to carry out the instruction.

- It is to be noted that we have kept the Flag register and the Stack Pointer (SP) in the decoder block, rather than in the register block.

- This is done in order to improve the performance of the decoder block so that it does not waste clock pulses in reading or updating the values of these registers from the register block in case of instructions like JNC (Jump Not Carry), ADD, etc.

- The decoder however does not have the accumulator, despite its frequent use, in order to safeguard its contents from being unintentionally changed during the many operations that it has to perform.

- Instead, the accumulator is equipped with four 16-bit registers, which act as intermediate storage for data during data shifting or manipulation operations like MOVE, ADD, AND etc.

- The decoder block, at first reads the values of the different registers involved in the operation from the register block, then, it stores these values in these intermediate registers.

- It then performs the required operations on them, and finally stores the result back in the register block, or in a memory or I/O block outside the microprocessor, as per the instructions.

- It is the only block that communicates with every other block.

- The interrupt signals coming from the interrupt block are used to interrupt the regular working of the decoder block.

- In case of an interrupt occurrence, the decoder block first finishes the task it was performing, and then it stores the contents of the current PC in the stack pointed by the stack pointer and then responds to the interrupt accordingly.

- It also sets (or resets) the EI (enable interrupt) flag, which is used to tell the interrupt block, if the decoder is willing to be interrupted or not.

- The format of the flag is as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	O	X	EI	S	Z	X	Ac	X	P	X	C

O – Overflow flag
EI – EI/DI flag
S – Sign flag, equal to the MSB of the result for signed operations
Z – Zero flag, is set if the result is zero, otherwise, it is reset.
Ac – Auxiliary Carry, it is set if a carry is generated from the lowest nibble.
P – Parity flag, is set if the result has even number of 1s, else, it is reset.
C – Carry flag, it is set if a carry is generated else it is reset, in case of subtraction it denotes borrow out.
X – Don't Care.

Register Block:

- The register block, as the name suggests, is a block of internal memory of the microprocessor. It consists of the accumulator, B, C, D, E, F registers and G, H, I register pairs.

- Among them, the accumulator, B, C, D, E , F are 16-bit registers and G, H and I are 32-bit register pairs, which can be divided into GH, GL, HH, HL, IH and IL registers (GH meaning the 16 higher order bits of the G register and GL the 16 lower order bits of the same).

- The register whose data the decoder block wants to access for read or write operation is decided by the address sent by the decoder through the *Reg. Sel. Line*.

- The register block is asynchronous as it does not have any input clock line, and is hence not depended on the system clock for its functioning. However, it can be reset by the *Reset In* line.

ALU:

- The ALU is the Arithmetic and Logical Unit of the microprocessor; in other words, it is the brain of the microprocessor, concerned with the arithmetic and logical operations performed by the microprocessor.

- It communicates only with the decoder block, has a parallel input/output, and is a combinational and asynchronous block.

- The decoder block gives the ALU the data(s) on which the operation is to be performed through the *Data A* (and *Data B*) line.

- The operation to be performed is mentioned to the ALU through the *Instr. Sel.* Line.

- The carry bit (or borrow) is entered in the ALU through the *Carry In* signal.

- The decoder block first loads these blocks with the required information and then enables the ALU using the *ALU Enable* signal.

- The ALU performs the required operations on the data provided and then gives the result as output in the *Result Data* line. The *Result Data* line is an 22-bit line, which contains the various flag values like carry, overload, etc.

- After the end of ALU operation, it enables the *ALU Proc. End* signal, which signals the decoder that the ALU operation is completed successfully. The ALU will not perform another operation unless the ALU Enable signal is deactivated and then again reactivated.

- This is to make sure that the ALU performs any operation only when the decoder wants it to perform the operation and only on those data send by it, also to safeguard against any change in the instruction or data midway through the operation due to interconnect losses.

Output Block:

- The output block is the block that is concerned with the writing of the data by the microprocessor in the external device, be it a memory or an I/O block.

- The data that is to be written can be either a data contained in one of the internal registers of the microprocessor, or it can be the result of some operation performed by the microprocessor.

- In order to write the data, the decoder at first writes the data in the *Output Data* line that is an input to the output block from the decoder.

- Then it writes the address of the memory or I/O to which the data needs to be written in the *Output Address* bus. This is the same bus as that of the precoder.

- Then the decoder informs if the output device is a memory or I/O block using the *IO/M Sel.* Line, which is also the same line that goes to the precoder block.

- The output block, writes these data from the *Output Data* bus and the *Output Address* bus to its internal output data register and output address register respectively. This it performs before the *Output Enable* signal goes HIGH.

VHDL Modeling:

a. Simulation Results:

In the following examples, we have simulated a few short programs in our microprocessor and have presented the simulation results along with explanations if needed.

Example 1.

Program statement:

Address	Mnemonics	Opcode
00000000	MVI A 01A9H	009D
00000001		01A9
00000002	PUSH A	00BF
00000003	NOP	0000

Program simulation:

Simulation explanation:

Here, we can see that at first the microprocessor is reset at 0ns. The clock duration for our microprocessor is 10ns, which makes it a 100MHz microprocessor. For the time period 0ns to 10ns, we see that the reset signal is LOW, which means that the microprocessor is being reset.

Then after 10ns, the reset signal is driven HIGH to allow the microprocessor to perform its operations. In the first clock pulse, i.e. from 10ns to 20ns, we see that the external data bus gives the 16 higher order bits of the address from which the microprocessor will begin its operation.

During this time, we can see that the ALE signal is driven HIGH, which demultiplexed the external data bus and stores the 16 higher order address bits in an address latch. The

external address bus on the other hand gives us the value of the 16 lower order address bits.

During 20ns to 60ns, the microprocessor reads the program from the external memory by incrementing the external address bus value by one and by using the read out signal, which is an active LOW signal, meaning that the data is read when the read out signal goes LOW. The program is first stored in the circular queue in the precoder block, thus implementing a pipelined flow of data.

In the duration 30ns to 120ns, the microprocessor processes the program and takes the appropriate steps and finally the output is given during the time interval 120ns to 140ns.

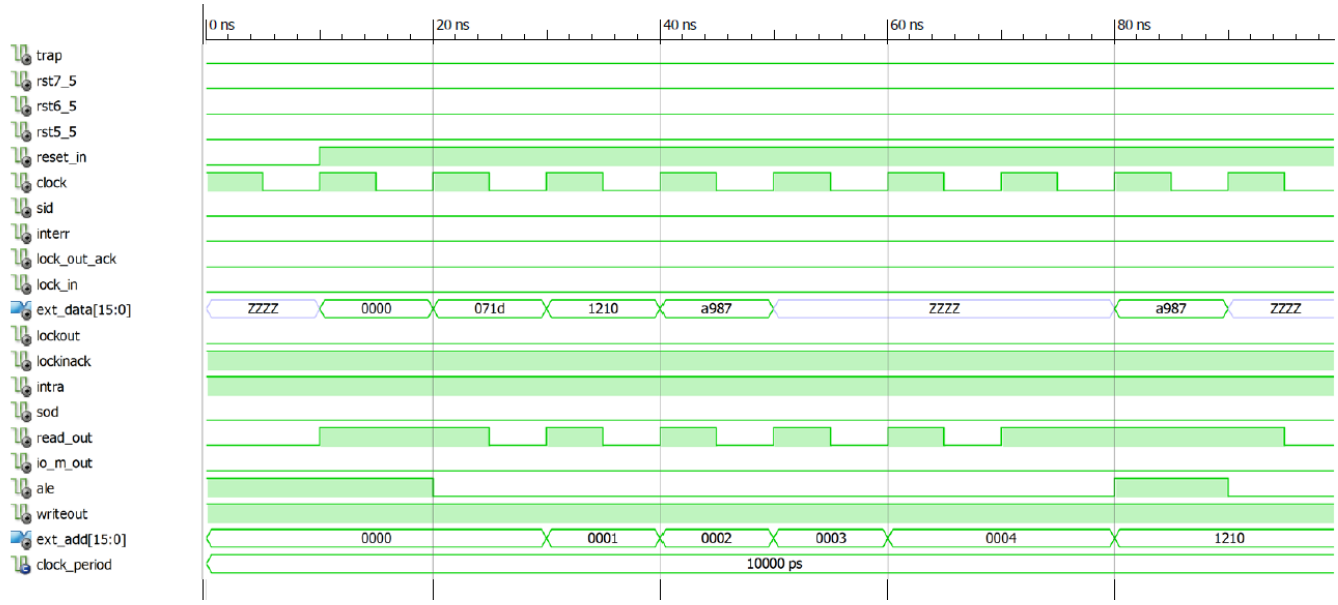
As we can see during this time interval, the value stred previously at accumulator by the command MVI A, is being pushed into the stack at the position $(FFFFFFFF)_{16}$. The method of pushing into the stack is quite similar to the initial read method, except that here instead of read out, the write out signal is activated, which is also an active LOW signal.

Example 2.

Program statement:

Address	**Mnemonics**	**Opcode**
00000000	JMP A9871210H	071D
00000001		1210
00000002		A987

Program simulation:



Simulation explanation:

Here we have simulated the execution of a JUMP command.

As we can see, at 10ns, the reset signal is driven HIGH to allow the microprocessor to perform its operations. In the first clock pulse, i.e. from 10ns to 20ns, we see that the

external data bus gives the 16 higher order bits of the address from which the microprocessor will begin its operation.

During 20ns to 60ns, the microprocessor reads the program from the external memory, and finds it to be a JUMP statement to a 32-bit address. The address is entered subsequent to the JUMP command. The 16 lower order address bits are entered first followed by the 16 higher order address bits.

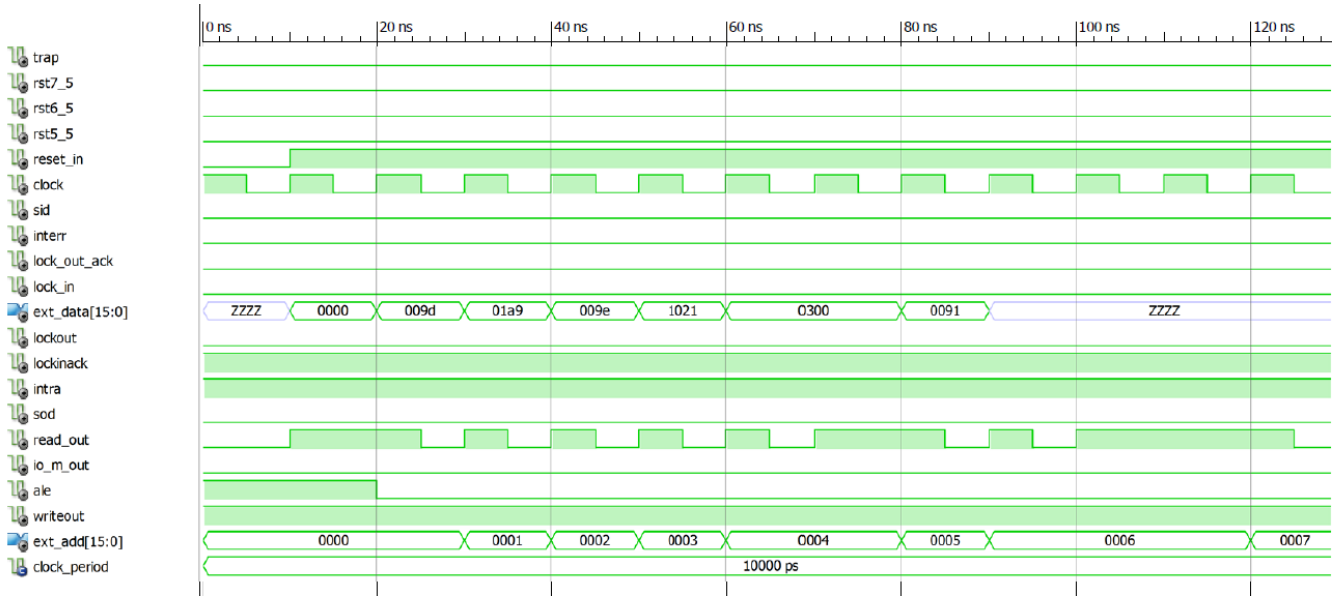
The microprocessor reads these values and the execution of the program JUMPs to that address after 80ns. It is to be noted that in this example an unconditional JUMP command is used, there are other commands that can JUMP the execution based on some condition, like the satus of a particular flag etc. In that case, if the condition is not satisfied, the JUMP does not occur.

Example 3.

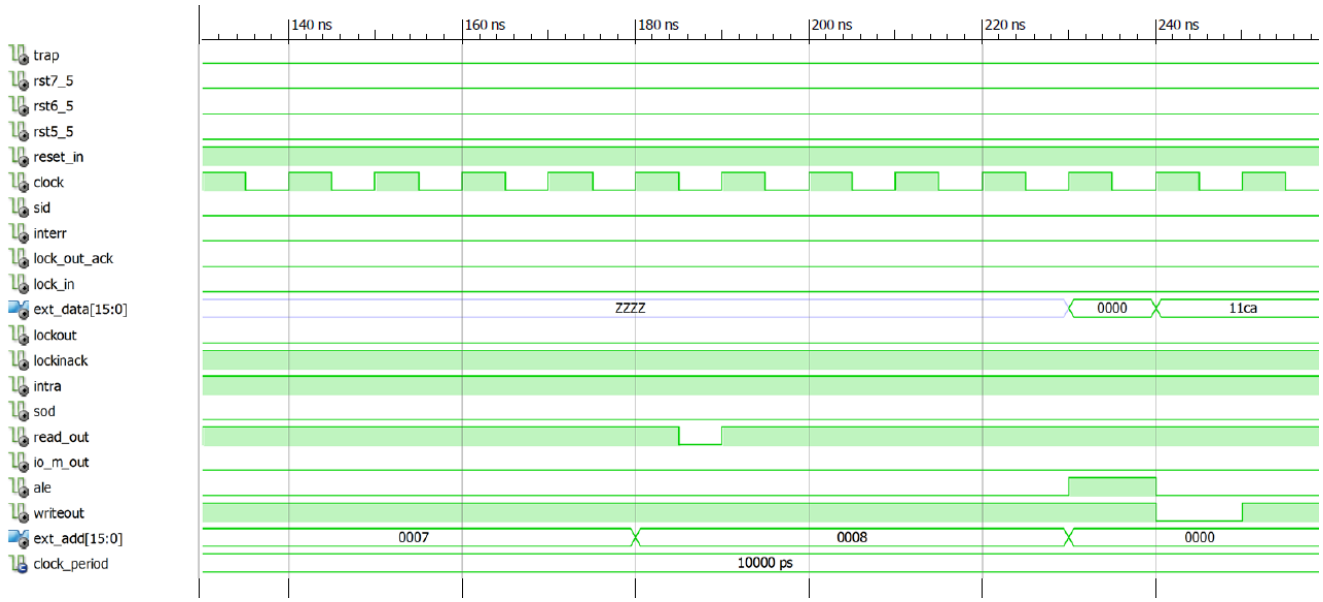
Program statement:

Address	Mnemonics	Opcode
00000000	MVI A 01A9H	009D
00000001		01A9
00000002	MVI B 1021H	009E
00000003		1021
00000004	ADD B	0300
00000005	MOV M,A	0091

Program simulation:



(contd.)



Simulation explanation:

Here we have simulated the execution of an ADD command and have shown the result using MOVE command.

At first, we load the accumulator and the B register of the microprocessor with the required values that need to be added, then we add accumulator to B register and the result is stored by default in the accumulator. This result is then read out by using the MOV M,A command, which copies the value stored at the accumulator to the address pointed by the value in the I register pair (also called as 'memory').

As we can see, during 230ns to 250ns, the result of the addition $(11CA)_{16}$, is written in $(00000000)_{16}$, which is the value of memory address.

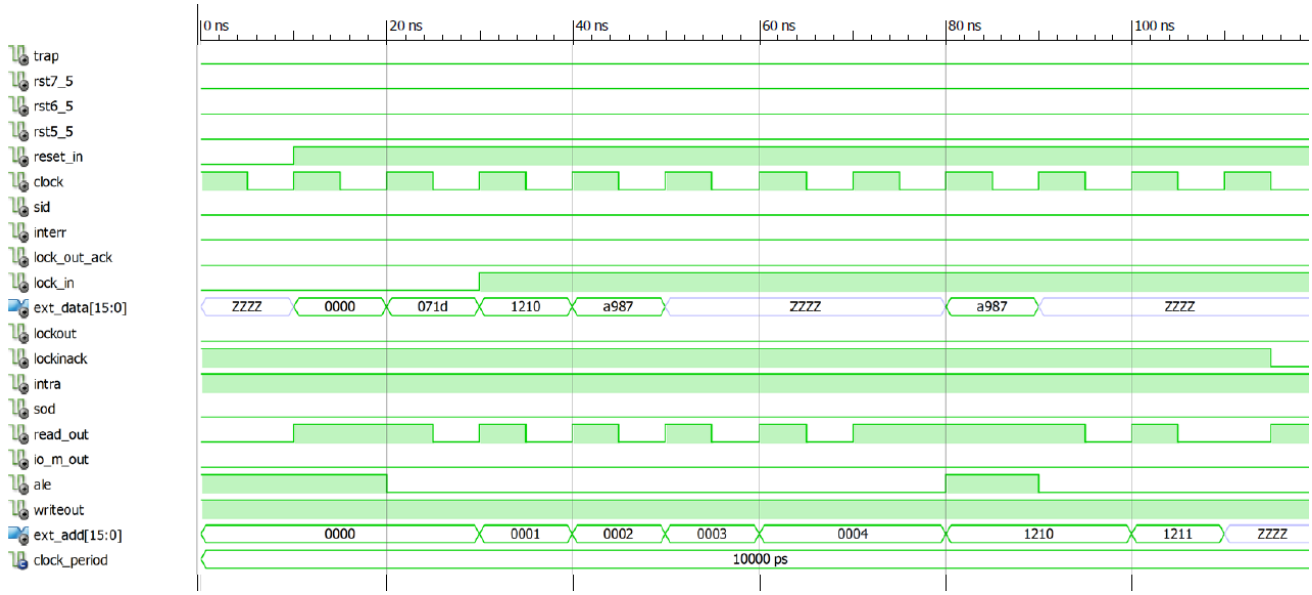
Example 4.

Program statement:

Address	**Mnemonics**	**Opcode**
00000000	JMP A9871210H	071D
00000001		1210
00000002		A987

The microprocessor is given a LOCK IN request from another microprocessor working in the same external bus at 30ns.

Program simulation:



Simulation explanation:

Here we have demonstrated what will happen if a LOCK IN signal occurs while the microprocessor is executing a command.

As we can see, at 20ns, the microprocessor is loaded with the JUMP command and it takes the corresponding actions as already explained in example 2. However, in this example, a LOCK IN request occurs from another microprocessor at 30ns.

In such a situation, as we can see, the microprocessor completes its current command execution and then hands over the bus to the other microprocessor by tri-stating the output of its own buses and signals the other microprocessor that it is ready for the handover by the help of the LOCK IN ACK. signal. The acknowledge signal is active LOW and goes LOW at 115ns as shown in the simulation result, after completion of the execution of the JUMP command.

It is to be noted that the LOCK IN signal is needed to be kept HIGH as long as the other microprocessor wishes to LOCK the shared bus for its own usage or else the current microprocessor will resume its operation as soon as the LOCK IN signal goes LOW.

The LOCK IN port is connected to the LOCK OUT port of the other microprocessor and the LOCK IN ACK. port is connected to the LOCK OUT ACK. port of the other microprocessor. The LOCK signal is generated using the LOCK command and it should be followed by a DLCK command when the bus is not needed to be locked any more.

Example 4.

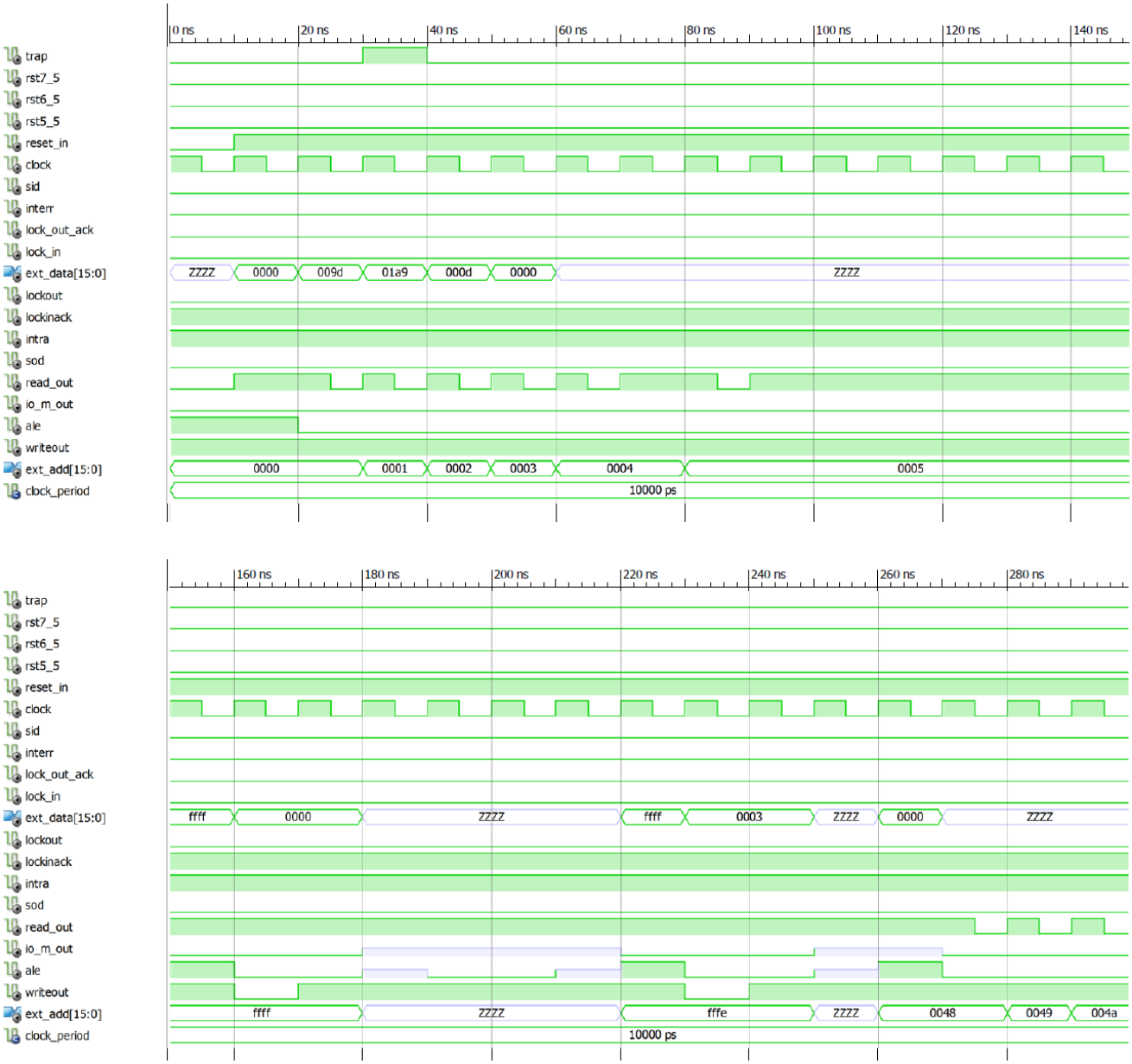
Program statement:

Address	**Mnemonics**	**Opcode**	
00000000	MVIA 01A9H	009D	
00000001		01A9	
00000002	MOV B,A	000D	(contd.)

Address	Mnemonics	Opcode
00000003	NOP	0000

The microprocessor is interrupted using the hardware interrupt TRAP at 30ns.

Program simulation:



Simulation explanation:

Here we have demonstrated what will happen if an interrupt occurs during the execution of the microprocessor. In our example, the microprocessor is interrupted using the hardware interrupt TRAP at 30ns, when it is in the middle of execution of the MVI A command.

Like the previous example, the microprocessor at first completes the execution of the current command and then performs a few housekeeping functions before transferring the execution to the address corresponding to TRAP.

For any interrupt, at first it pushes the current program counter (PC) values into the stack as shown in the time period 150ns to 250ns. It is to be noted that it pushes the 16 higher order bits of the address first and then pushes the 16 lower order bits into the stack.

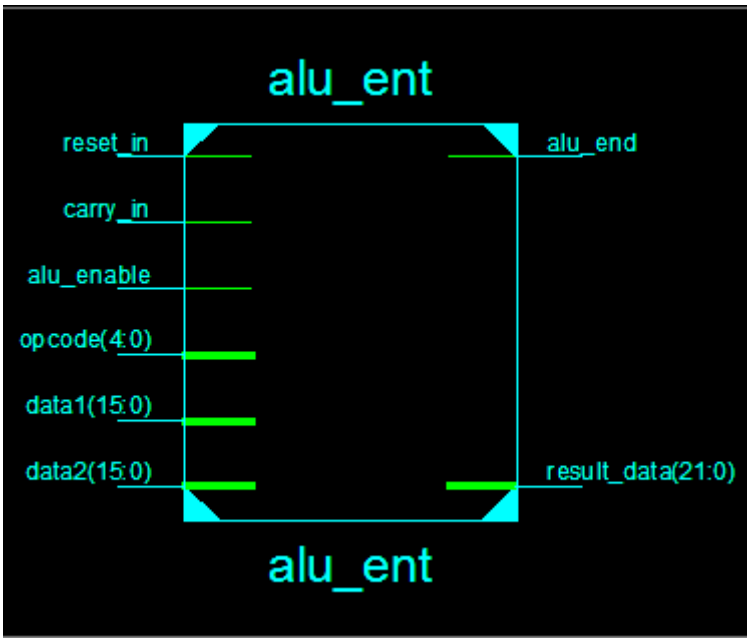
Also note that even though the microprocessor had previously stored the subsequent commands in the pipeline, the PC contains the values of the address corresponding to the opcode which is currently being processed.

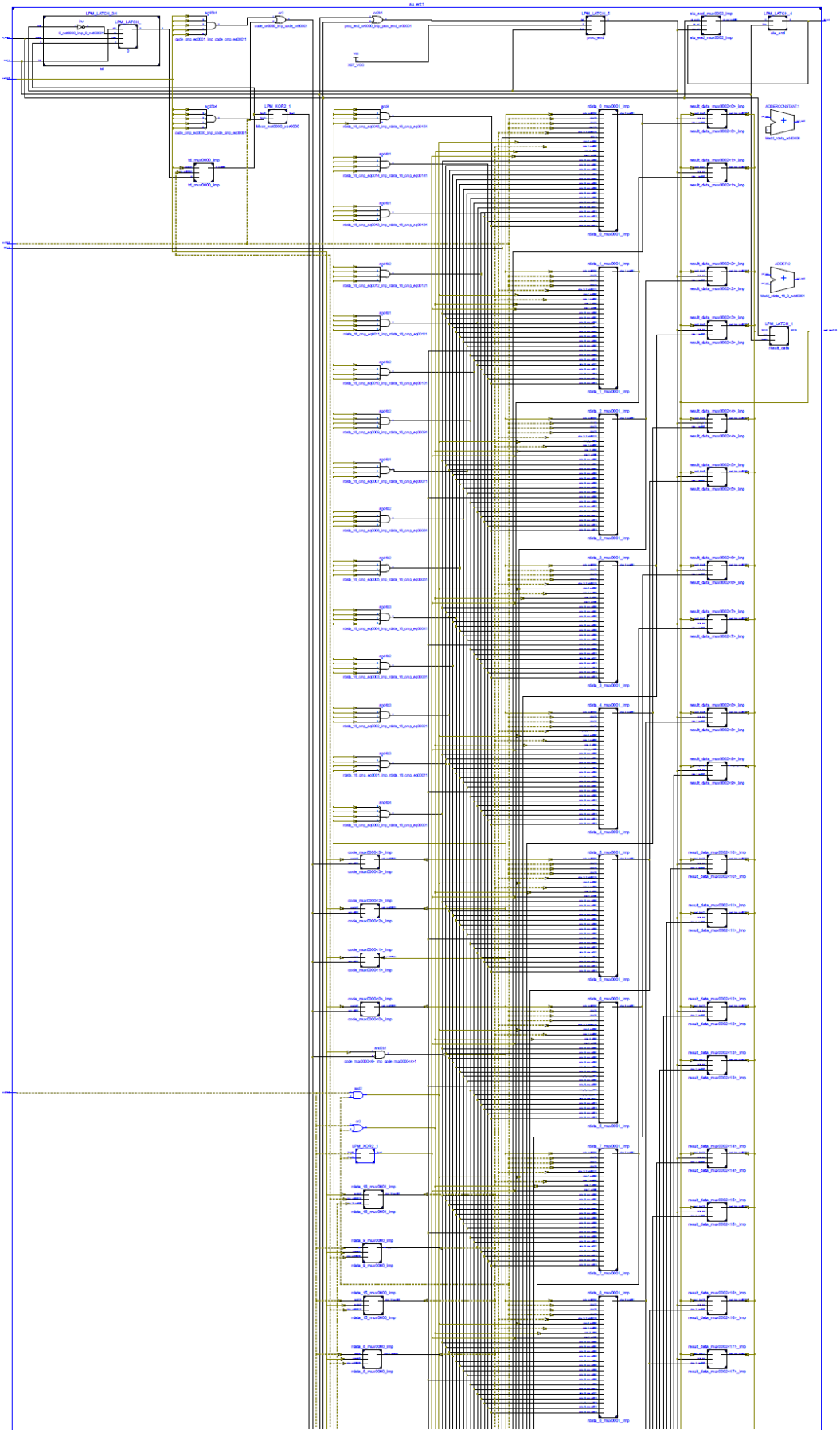
To return to the execution of the current program, the interrupt subroutine must have a RET (return) statement. It is also to be noted that none of the register contents are pushed into the stack. This means that the values stored in these registers can be modified in the subroutine. Care should be taken to restore their original values before returning from the subroutine.

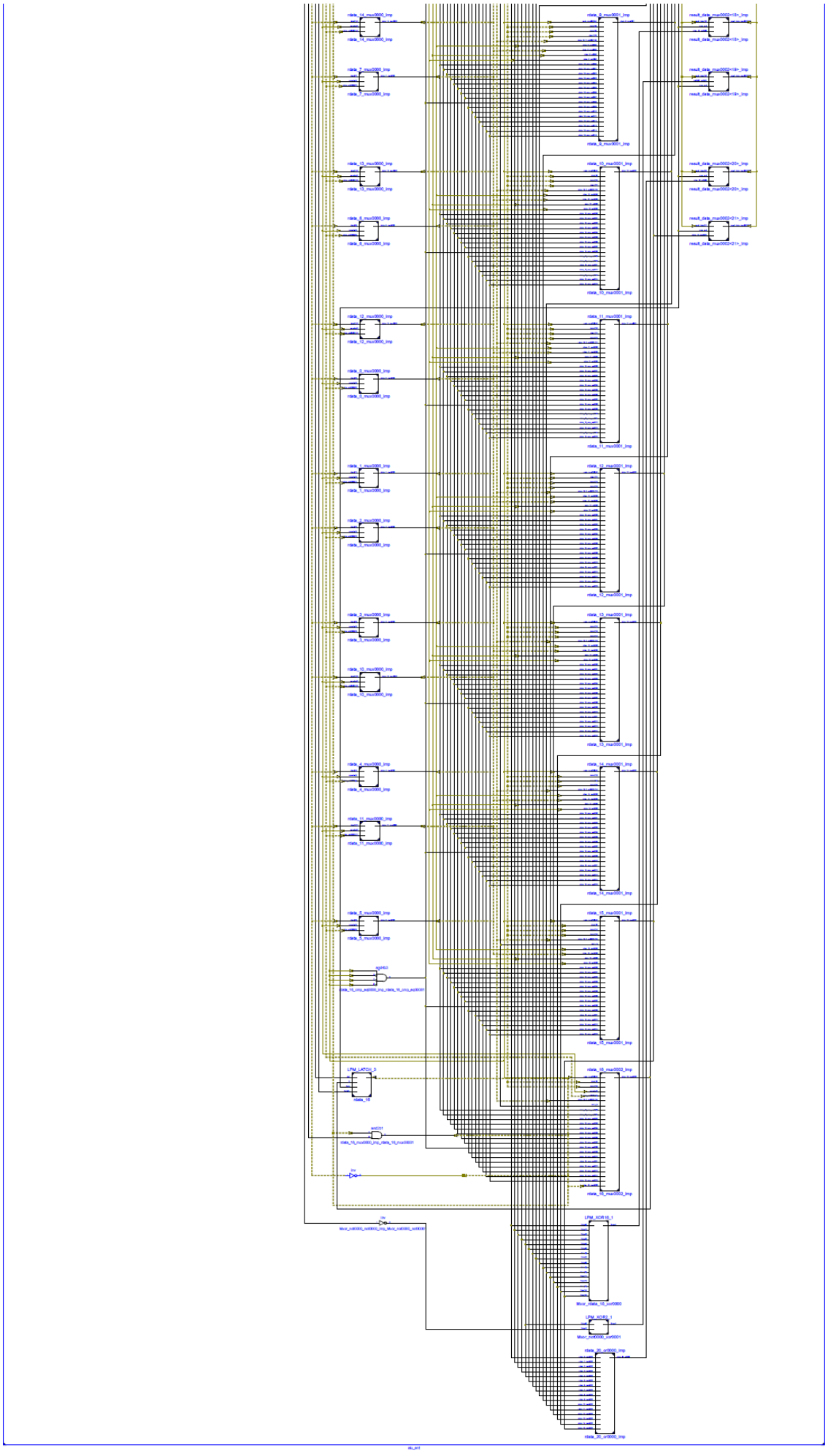
b. RTL Schematic:

Below are the RTL schematics generated for a few of the blocks, due to size constraints we have not presented the schematics of a few blocks.

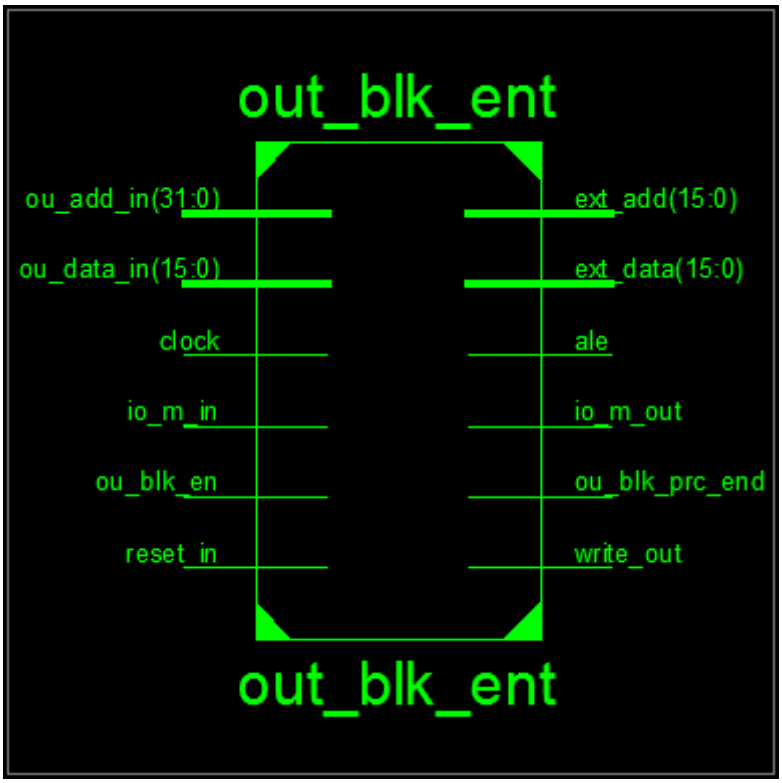
ALU Block:

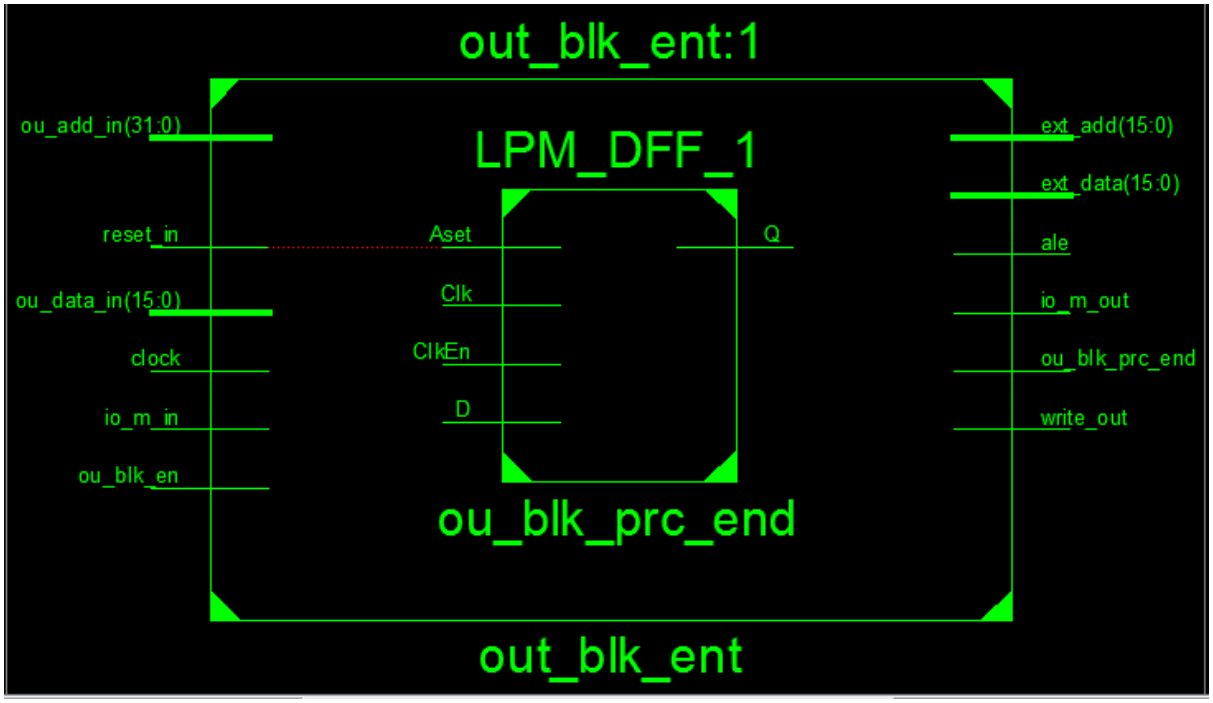


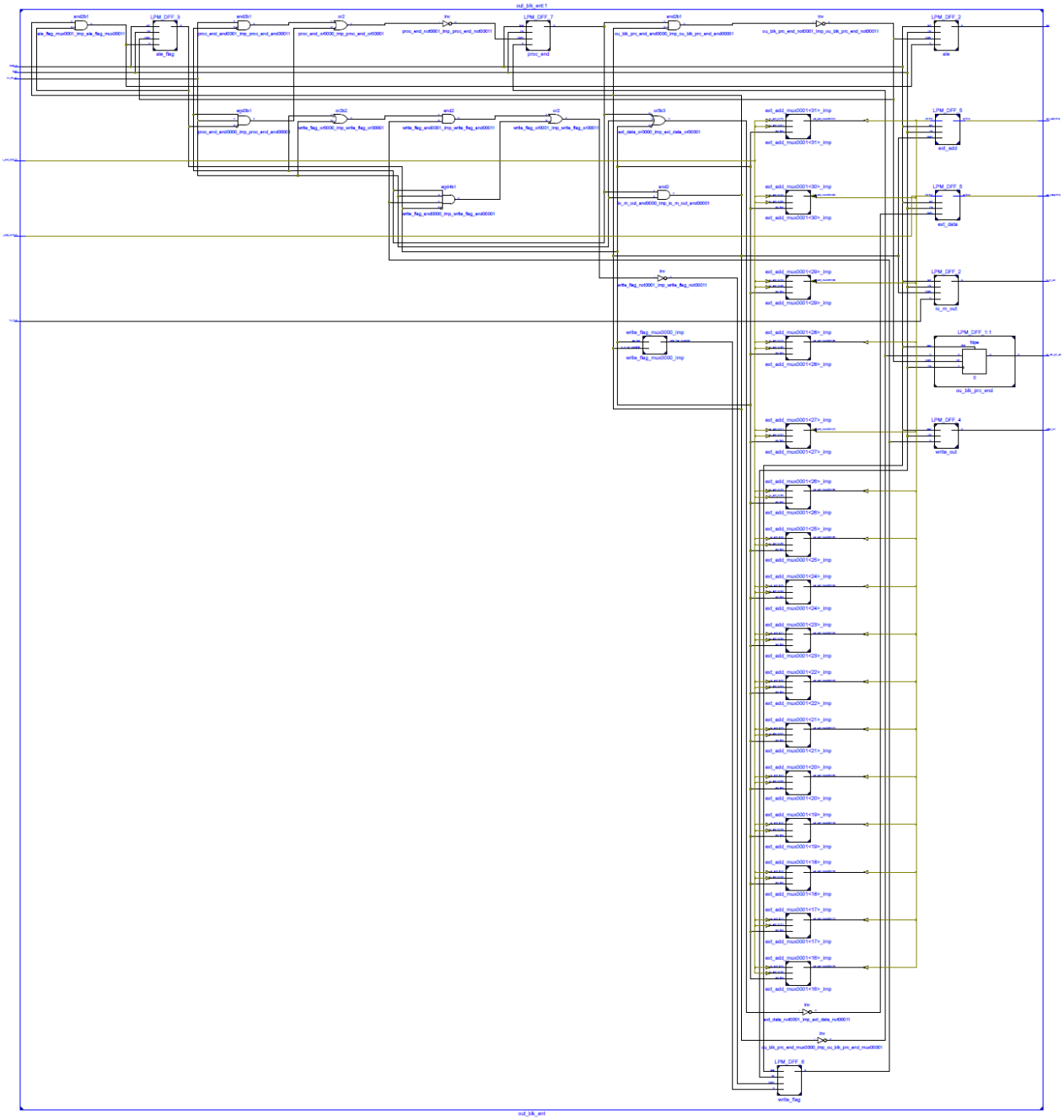




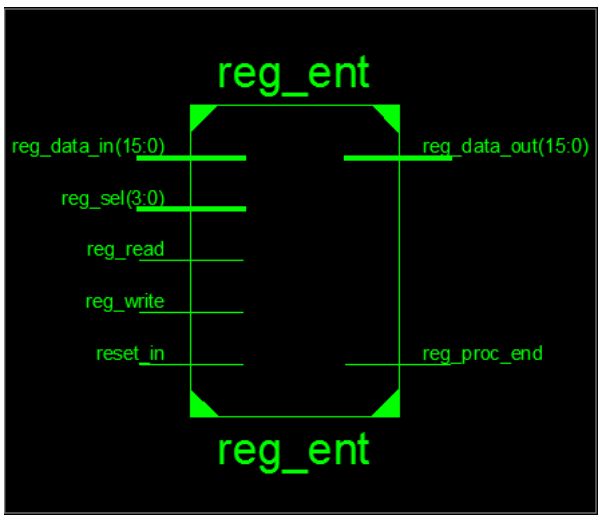
Output Block:

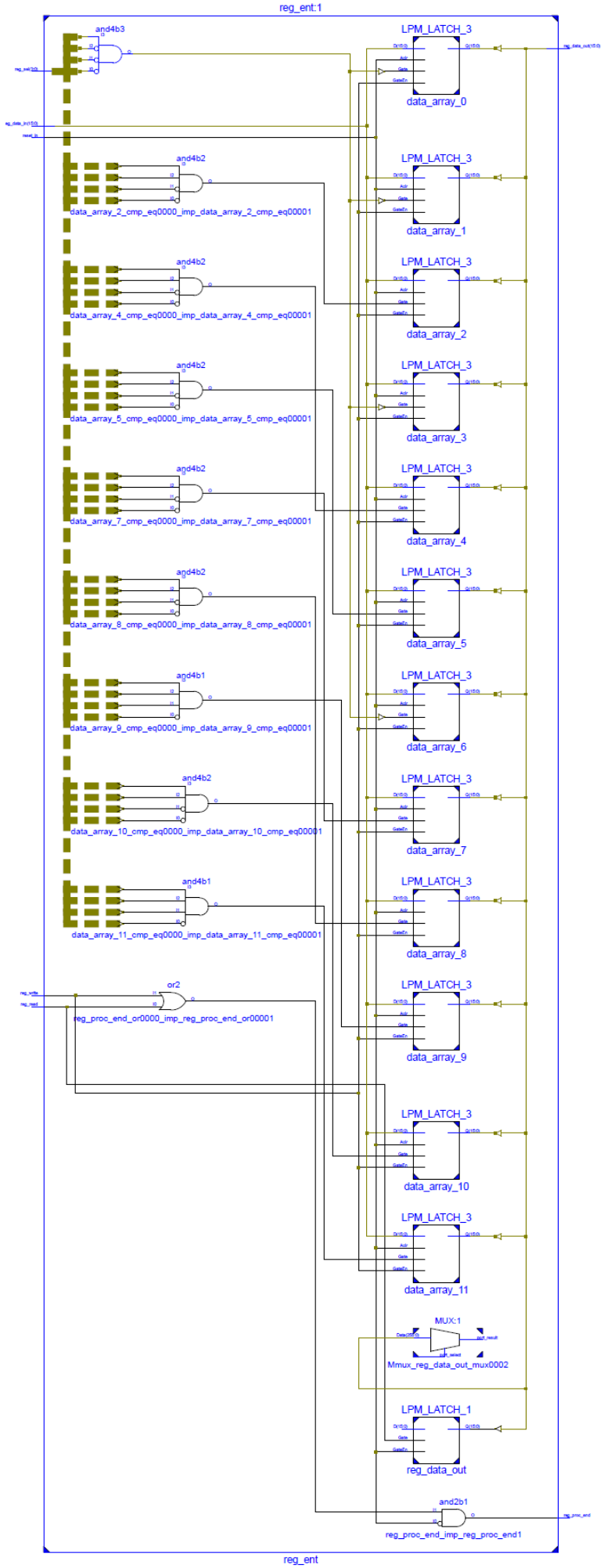




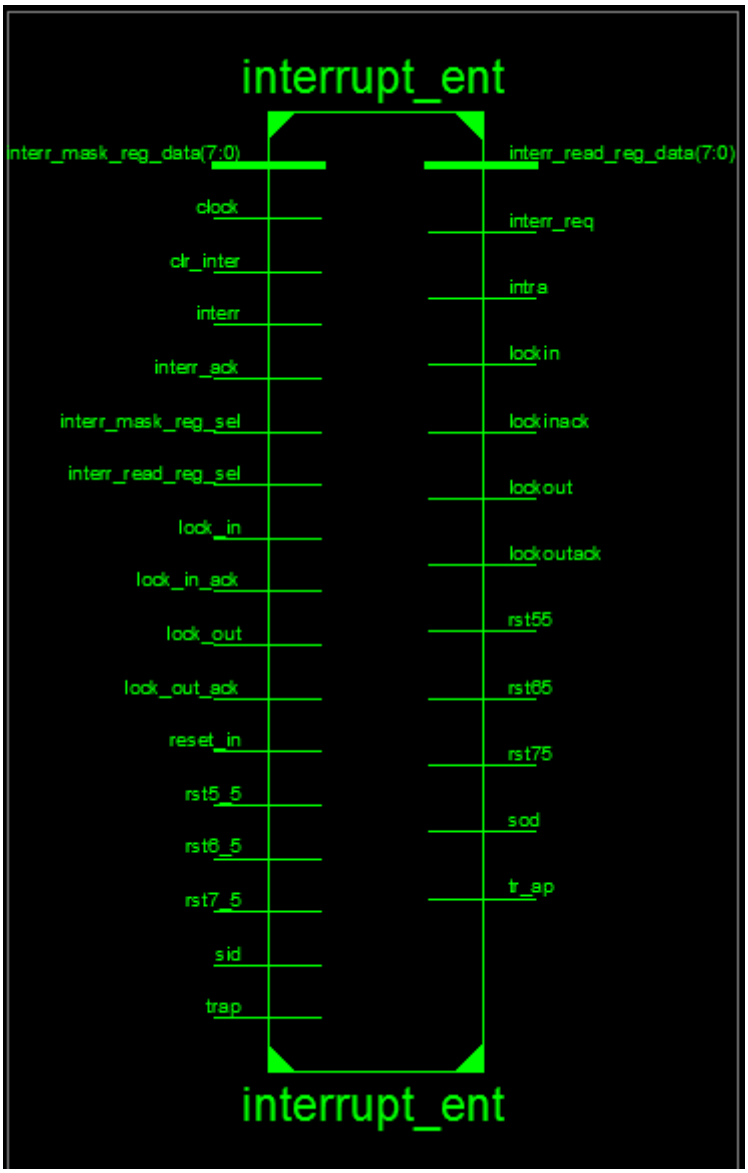


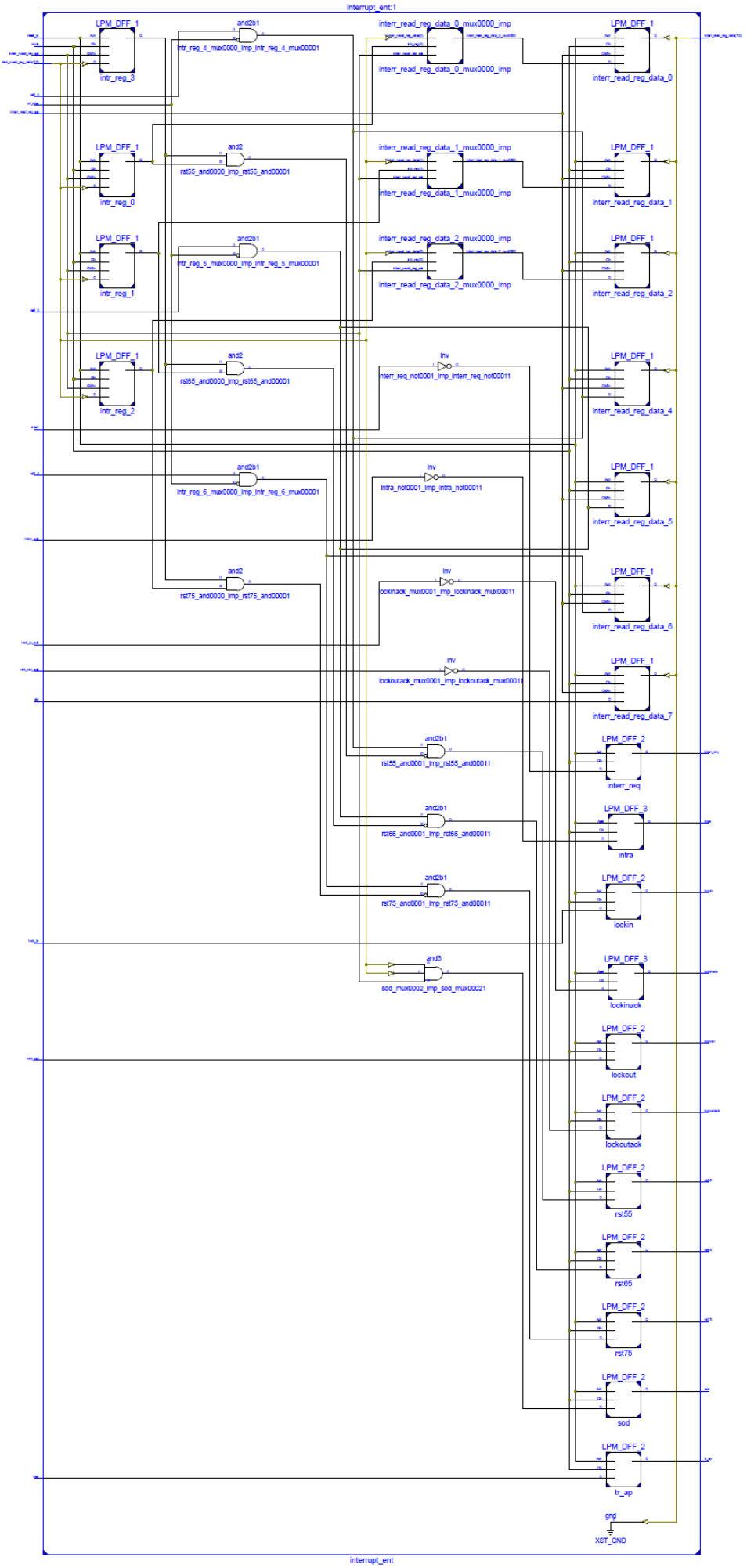
Register Block:



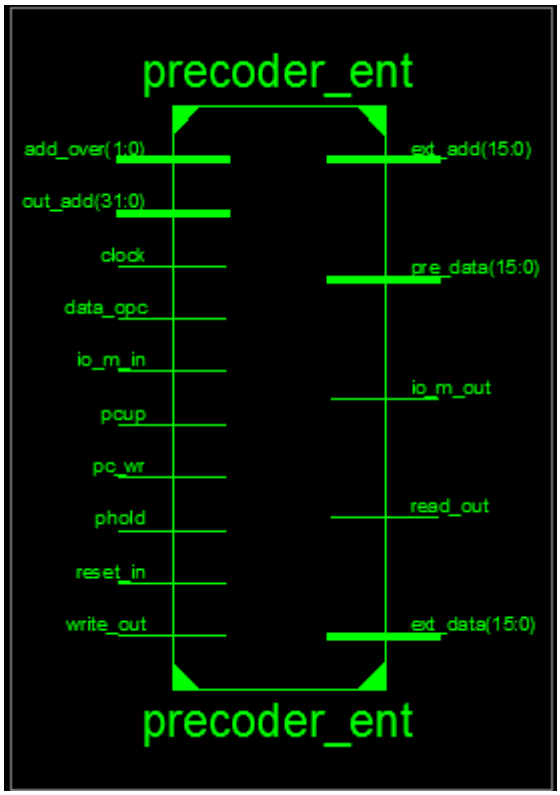

reg_ent:1
and4b3
LPM_LATCH_3
data_array_0
and4b2
data_array_2_cmp_eq0000_imp_data_array_2_cmp_eq00001
LPM_LATCH_3
data_array_1
and4b2
data_array_4_cmp_eq0000_imp_data_array_4_cmp_eq00001
LPM_LATCH_3
data_array_2
and4b2
data_array_5_cmp_eq0000_imp_data_array_5_cmp_eq00001
LPM_LATCH_3
data_array_3
and4b2
data_array_7_cmp_eq0000_imp_data_array_7_cmp_eq00001
LPM_LATCH_3
data_array_4
and4b2
data_array_8_cmp_eq0000_imp_data_array_8_cmp_eq00001
LPM_LATCH_3
data_array_5
and4b1
data_array_9_cmp_eq0000_imp_data_array_9_cmp_eq00001
LPM_LATCH_3
data_array_6
and4b2
data_array_10_cmp_eq0000_imp_data_array_10_cmp_eq00001
LPM_LATCH_3
data_array_7
and4b1
data_array_11_cmp_eq0000_imp_data_array_11_cmp_eq00001
LPM_LATCH_3
data_array_8
or2
reg_proc_end_or0000_imp_reg_proc_end_or00001
LPM_LATCH_3
data_array_9
LPM_LATCH_3
data_array_10
LPM_LATCH_3
data_array_11
MUX:1
Mmux_reg_data_out_mux0002
LPM_LATCH_1
reg_data_out
and2b1
reg_proc_end_imp_reg_proc_end1
reg_ent

Interrupt Block:




interrupt_ent:1
LPM_DFF_1
intr_reg_3
and2b1
intr_reg_4_mux0000_lmp_intr_reg_4_mux00001
interr_read_reg_data_0_mux0000_imp
interr_read_reg_data_0_mux0000_imp
LPM_DFF_1
interr_read_reg_data_0
LPM_DFF_1
intr_reg_0
and2
rst55_and0000_lmp_rst55_and00001
interr_read_reg_data_1_mux0000_imp
interr_read_reg_data_1_mux0000_imp
LPM_DFF_1
interr_read_reg_data_1
LPM_DFF_1
intr_reg_1
and2b1
intr_reg_5_mux0000_lmp_intr_reg_5_mux00001
interr_read_reg_data_2_mux0000_imp
interr_read_reg_data_2_mux0000_imp
LPM_DFF_1
interr_read_reg_data_2
LPM_DFF_1
intr_reg_2
and2
rst65_and0000_lmp_rst65_and00001
inv
interr_req_not0001_lmp_interr_req_not00011
LPM_DFF_1
interr_read_reg_data_4
and2b1
intr_reg_6_mux0000_lmp_intr_reg_6_mux00001
inv
intra_not0001_lmp_intra_not00011
LPM_DFF_1
interr_read_reg_data_5
and2
rst75_and0000_lmp_rst75_and00001
inv
lockinack_mux0001_lmp_lockinack_mux00011
LPM_DFF_1
interr_read_reg_data_6
inv
lockoutack_mux0001_lmp_lockoutack_mux00011
LPM_DFF_1
interr_read_reg_data_7
and2b1
rst55_and0001_lmp_rst55_and00011
LPM_DFF_2
interr_req
and2b1
rst65_and0001_lmp_rst65_and00011
LPM_DFF_3
intra
and2b1
rst75_and0001_lmp_rst75_and00011
LPM_DFF_2
lockin
and3
sod_mux0002_lmp_sod_mux00021
LPM_DFF_3
lockinack
LPM_DFF_2
lockout
LPM_DFF_2
lockoutack
LPM_DFF_2
rst55
LPM_DFF_2
rst65
LPM_DFF_2
rst75
LPM_DFF_2
sod
LPM_DFF_2
tr_ap
gnd
XST_GND
interrupt_ent

Precoder Block:



Future Plans:

We have, as of yet, successfully coded and simulated the entire microprocessor. In the future, we look forward to expanding the opcode list to include those opcodes that we had to ignore because of time constraints.

We also look forward to increasing the clock speed of our microprocessor thereby making it faster and more useful practically.

We believe that our microprocessor is as efficient as any other general-purpose microprocessor like Intel 8085 and will be able to perform all the tasks that any other general-purpose microprocessors can perform.

References:

1. “Microprocessor Architecture, Programming, and Applications with the 8085”, Fifth Edition, Ramesh Gaonkar.

2. “VHDL Coding and Logic Synthesis with Synopsys ®”, Weng Fook Lee.

3. “A VHDL Primer”, Jayaram Bhasker.

4. “Circuit Design with VHDL”, Volnei A. Pedroni.

5. “Digital Design: An Embedded Systems Approach Using VHDL”, Peter J. Ashenden.

6. “http://esd.cs.ucr.edu/labs/tutorial/” “VHDL Tutorial: Learn by Example”, by Weijun Zhang.

7. “Digital Systems Design Using VHDL®” Charles H. Roth, Jr.

8. “The Designer’s Guide to VHDL”, Third Edition, Peter J. Ashenden & Jim Lewis.

9. “VHDL 101 Everything you need to know to get started”, William Kafig.

10. “Advanced Microprocessors and Peripherals”, Second Edition, A. K. Ray & K. M. Bhurchandi.

11. “A Report on Practical Training Taken at Linux Soft Technologies PVT. LTD., Jaipur”, Ankur Bhatt (Roll No.-060207, Session- 2009-10, Dept. of Electronics & Communication Engineering, Government Engineering College Ajmer).

12. “http://www.seas.upenn.edu/~ese171/vhdl/vhdl_primer.html#_Toc526061342”, VHDL Tutorial by Jan Van der Spiegel, University of Pennsylvania, Department of Electrical and Systems Engineering.

APPENDIX – A OPCODE LIST

Opcode	Mnemonics	Comments
0000	NOP	No Operation
0001	MOV A,B	Move the contents of B to A.
0002	MOV A,C	Move the contents of C to A.
0003	MOV A,D	Move the contents of D to A.
0004	MOV A,E	Move the contents of E to A.
0005	MOV A,F	Move the contents of F to A.
0006	MOV A,M	Move the contents from memory to A.
0007	MOV A,GH	Move the contents of 16 higher order bits of G to A.
0008	MOV A,GL	Move the contents of 16 lower order bits of G to A.
0009	MOV A,HH	Move the contents of 16 higher order bits of H to A.
000A	MOV A,HL	Move the contents of 16 lower order bits of H to A.
000B	MOV A,IH	Move the contents of 16 higher order bits of I to A.
000C	MOV A,IL	Move the contents of 16 lower order bits of I to A.
000D	MOV B,A	Move the contents of A to B.
000E	MOV B,C	Move the contents of C to B.
000F	MOV B,D	Move the contents of C to B.
0010	MOV B,E	Move the contents of C to B.
0011	MOV B,F	Move the contents of C to B.
0012	MOV B,M	Move the contents of memory to B.
0013	MOV B,GH	Move the contents of 16 higher order bits of G to B.
0014	MOV B,GL	Move the contents of 16 lower order bits of G to B.
0015	MOV B,HH	Move the contents of 16 higher order bits of H to B.
0016	MOV B,HL	Move the contents of 16 lower order bits of H to B.
0017	MOV B,IH	Move the contents of 16 higher order bits of I to B.
0018	MOV B,IL	Move the contents of 16 lower order bits of I to B.
0019	MOV C,A	Move the contents of A to C.
001A	MOV C,B	Move the contents of B to C.
001B	MOV C,D	Move the contents of D to C.
001C	MOV C,E	Move the contents of E to C.
001D	MOV C,F	Move the contents of F to C.
001E	MOV C,M	Move the contents of memory to C.
001F	MOV C,GH	Move the contents of 16 higher order bits of G to C.
0020	MOV C,GL	Move the contents of 16 lower order bits of G to C.
0021	MOV C,HH	Move the contents of 16 higher order bits of H to C.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
0022	MOV C,HL	Move the contents of 16 lower order bits of H to C.
0023	MOV C,IH	Move the contents of 16 higher order bits of I to C.
0024	MOV C,IL	Move the contents of 16 lower order bits of I to C.
0025	MOV D,A	Move the contents of A to D.
0026	MOV D,B	Move the contents of B to D.
0027	MOV D,C	Move the contents of C to D.
0028	MOV D,E	Move the contents of E to D.
0029	MOV D,F	Move the contents of F to D.
002A	MOV D,M	Move the contents of memory to D.
002B	MOV D,GH	Move the contents of 16 higher order bits of G to D.
002C	MOV D,GL	Move the contents of 16 lower order bits of G to D.
002D	MOV D,HH	Move the contents of 16 higher order bits of H to D.
002E	MOV D,HL	Move the contents of 16 lower order bits of H to D.
002F	MOV D,IH	Move the contents of 16 higher order bits of I to D.
0030	MOV D,IL	Move the contents of 16 lower order bits of I to C.
0031	MOV E,A	Move the contents of A to E.
0032	MOV E,B	Move the contents of B to E.
0033	MOV E,C	Move the contents of C to E.
0034	MOV E,D	Move the contents of D to E.
0035	MOV E,F	Move the contents of F to E.
0036	MOV E,M	Move the contents of memory to E.
0037	MOV E,GH	Move the contents of 16 higher order bits of G to E.
0038	MOV E,GL	Move the contents of 16 lower order bits of G to E.
0039	MOV E,HH	Move the contents of 16 higher order bits of H to E.
003A	MOV E,HL	Move the contents of 16 lower order bits of H to E.
003B	MOV E,IH	Move the contents of 16 higher order bits of I to E.
003C	MOV E,IL	Move the contents of 16 lower order bits of I to E.
003D	MOV F,A	Move the contents of A to F.
003E	MOV F,B	Move the contents of B to F.
003F	MOV F,C	Move the contents of C to F.
0040	MOV F,D	Move the contents of D to F.
0041	MOV F,E	Move the contents of E to F.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
0042	MOV F,M	Move the contents of memory to F.
0043	MOV F,GH	Move the contents of 16 higher order bits of G to F.
0044	MOV F,GL	Move the contents of 16 lower order bits of G to F.
0045	MOV F,HH	Move the contents of 16 higher order bits of H to F.
0046	MOV F,HL	Move the contents of 16 lower order bits of H to F.
0047	MOV F,IH	Move the contents of 16 higher order bits of I to F.
0048	MOV F,IL	Move the contents of 16 lower order bits of I to F.
0049	MOV GH,A	Move the contents of A to 16 higher order bits of G.
004A	MOV GH,B	Move the contents of B to 16 higher order bits of G.
004B	MOV GH,C	Move the contents of C to 16 higher order bits of G.
004C	MOV GH,D	Move the contents of D to 16 higher order bits of G.
004D	MOV GH,E	Move the contents of E to 16 higher order bits of G.
004E	MOV GH,F	Move the contents of F to 16 higher order bits of G.
004F	MOV GH,M	Move the contents of memory to 16 higher order bits of G.
0050	MOV GH,GL	Move the contents of 16 lower order bits of G to 16 higher order bits of G.
0051	MOV GH,HH	Move the contents of 16 higher order bits of H to 16 higher order bits of G.
0052	MOV GH,HL	Move the contents of 16 lower order bits of H to 16 higher order bits of G.
0053	MOV GH,IH	Move the contents of 16 higher order bits of I to 16 higher order bits of G.
0054	MOV GH,IL	Move the contents of 16 lower order bits of I to 16 higher order bits of G.
0055	MOV GL,A	Move the contents of A to 16 lower order bits of G.
0056	MOV GL,B	Move the contents of B to 16 lower order bits of G.
0057	MOV GL,C	Move the contents of C to 16 lower order bits of G.
0058	MOV GL,D	Move the contents of D to 16 lower order bits of G.
0059	MOV GL,E	Move the contents of E to 16 lower order bits of G.
005A	MOV GL,F	Move the contents of F to 16 lower order bits of G.
005B	MOV GL,M	Move the contents of memory to 16 lower order bits of G.
005C	MOV GL,GH	Move the contents of 16 higher order bits of G to 16 lower order bits of G.
005D	MOV GL,HH	Move the contents of 16 higher order bits of H to 16 lower order bits of G.
005E	MOV GL,HL	Move the contents of 16 lower order bits of H to 16 lower order bits of G.
005F	MOV GL,IH	Move the contents of 16 higher order bits of I to 16 lower order bits of G.
0060	MOV GL,IL	Move the contents of 16 lower order bits of I to 16 lower order bits of G.
0061	MOV HH,A	Move the contents of A to 16 higher order bits of H.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
0062	MOV HH,B	Move the contents of B to 16 higher order bits of H.
0063	MOV HH,C	Move the contents of C to 16 higher order bits of H.
0064	MOV HH,D	Move the contents of D to 16 higher order bits of H.
0065	MOV HH,E	Move the contents of E to 16 higher order bits of H.
0066	MOV HH,F	Move the contents of F to 16 higher order bits of H.
0067	MOV HH,M	Move the contents of memory to 16 higher order bits of H.
0068	MOV HH,GH	Move the contents of 16 higher order bits of G to 16 higher order bits of H.
0069	MOV HH,GL	Move the contents of 16 lower order bits of G to 16 higher order bits of H.
006A	MOV HH,HL	Move the contents of 16 lower order bits of H to 16 higher order bits of H.
006B	MOV HH,IH	Move the contents of 16 higher order bits of I to 16 higher order bits of H.
006C	MOV HH,IL	Move the contents of 16 lower order bits of I to 16 higher order bits of H.
006D	MOV HL,A	Move the contents of A to 16 lower order bits of H.
006E	MOV HL,B	Move the contents of B to 16 lower order bits of H.
006F	MOV HL,C	Move the contents of C to 16 lower order bits of H.
0070	MOV HL,D	Move the contents of D to 16 lower order bits of H.
0071	MOV HL,E	Move the contents of E to 16 lower order bits of H.
0072	MOV HL,F	Move the contents of F to 16 lower order bits of H.
0073	MOV HL,M	Move the contents of memory to 16 lower order bits of H.
0074	MOV HL,GH	Move the contents of 16 higher order bits of G to 16 lower order bits of H.
0075	MOV HL,GL	Move the contents of 16 lower order bits of G to 16 lower order bits of H.
0076	MOV HL,HH	Move the contents of 16 higher order bits of H to 16 lower order bits of H.
0077	MOV HL,IH	Move the contents of 16 higher order bits of I to 16 lower order bits of H.
0078	MOV HL,IL	Move the contents of 16 lower order bits of I to 16 lower order bits of H.
0079	MOV IH,A	Move the contents of A to 16 higher order bits of I.
007A	MOV IH,B	Move the contents of B to 16 higher order bits of I.
007B	MOV IH,C	Move the contents of C to 16 higher order bits of I.
007C	MOV IH,D	Move the contents of D to 16 higher order bits of I.
007D	MOV IH,E	Move the contents of E to 16 higher order bits of I.
007E	MOV IH,F	Move the contents of F to 16 higher order bits of I.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
007F	MOV IH,M	Move the contents of memory to 16 higher order bits of I.
0080	MOV IH,GH	Move the contents of 16 higher order bits of G to 16 higher order bits of I.
0081	MOV IH,GL	Move the contents of 16 lower order bits of G to 16 higher order bits of I.
0082	MOV IH,HH	Move the contents of 16 higher order bits of H to 16 higher order bits of I.
0083	MOV IH,HL	Move the contents of 16 lower order bits of H to 16 higher order bits of I.
0084	MOV IH,IL	Move the contents of 16 lower order bits of I to 16 higher order bits of I.
0085	MOV IL,A	Move the contents of A to 16 lower order bits of I.
0086	MOV IL,B	Move the contents of B to 16 lower order bits of I.
0087	MOV IL,C	Move the contents of C to 16 lower order bits of I.
0088	MOV IL,D	Move the contents of D to 16 lower order bits of I.
0089	MOV IL,E	Move the contents of E to 16 lower order bits of I.
008A	MOV IL,F	Move the contents of F to 16 lower order bits of I.
008B	MOV IL,M	Move the contents of memory to 16 lower order bits of I.
008C	MOV IL,GH	Move the contents of 16 higher order bits of G to 16 lower order bits of I.
008D	MOV IL,GL	Move the contents of 16 lower order bits of G to 16 lower order bits of I.
008E	MOV IL,HH	Move the contents of 16 higher order bits of H to 16 lower order bits of I.
008F	MOV IL,HL	Move the contents of 16 lower order bits of H to 16 lower order bits of I.
0090	MOV IL,IH	Move the contents of 16 higher order bits of I to 16 lower order bits of I.
0091	MOV M,A	Move the contents of A to memory.
0092	MOV M,B	Move the contents of B to memory.
0093	MOV M,C	Move the contents of C to memory.
0094	MOV M,D	Move the contents of D to memory.
0095	MOV M,E	Move the contents of E to memory.
0096	MOV M,F	Move the contents of F to memory.
0097	MOV M,GH	Move the contents of 16 higher order bits of G to memory.
0098	MOV M,GL	Move the contents of 16 lower order bits of G to memory.
0099	MOV M,HH	Move the contents of 16 higher order bits of H to memory.
009A	MOV M,HL	Move the contents of 16 lower order bits of H to memory.
009B	MOV M,IH	Move the contents of 16 higher order bits of I to memory.
009C	MOV M,IL	Move the contents of 16 lower order bits of I to memory.
009D	MVI A,data	Move immediate 16-bit data to A.
009E	MVI B,data	Move immediate 16-bit data to B.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
009F	MVI C,data	Move immediate 16-bit data to C.
00A0	MVI D,data	Move immediate 16-bit data to D.
00A1	MVI E,data	Move immediate 16-bit data to E.
00A2	MVI F,data	Move immediate 16-bit data to F.
00A3	MVI GH,data	Move immediate 16-bit data to 16 higher order bits of G.
00A4	MVI GL,data	Move immediate 16-bit data to 16 lower order bits of G.
00A5	MVI HH,data	Move immediate 16-bit data to 16 higher order bits of H.
00A6	MVI HL,data	Move immediate 16-bit data to 16 lower order bits of H.
00A7	MVI IH,data	Move immediate 16-bit data to 16 higher order bits of I.
00A8	MVI IL,data	Move immediate 16-bit data to 16 lower order bits of I.
00A9	LDA address	Load the contents at the memory address directly to the accumulator.
00AA	LDAX C	Load the contents at the memory address pointed by the value stored in the C/D register pair directly to the accumulator.
00AB	LDAX E	Load the contents at the memory address pointed by the value stored in the E/F register pair directly to the accumulator.
00AC	LDAX G	Load the contents at the memory address pointed by the value stored in the G register directly to the accumulator.
00AD	LDAX H	Load the contents at the memory address pointed by the value stored in the H register directly to the accumulator.
00AE	LXI C,data	Loads the data directly to the C/D register pair.
00AF	LXI E,data	Loads the data directly to the E/F register pair.
00B0	LXI G,data	Loads the data directly to G.
00B1	LXI H,data	Loads the data directly to H.
00B2	LXI I,data	Loads the data directly to I.
00B3	LXI SP,data	Loads the data directly to SP.
00B4	MOVS length	Moves the string of data of the given length, starting from memory location pointed by I to the memory locations staring from that pointed by H.
00B5	POP A	POPs the data stored in the stack pointed by the SP and stores it at the Accumulator.
00B6	POP B	POPs the data stored in the stack pointed by the SP and stores it at B.
00B7	POP C	POPs the data stored in the stack pointed by the SP and stores it at C.
00B8	POP D	POPs the data stored in the stack pointed by the SP and stores it at D.
00B9	POP E	POPs the data stored in the stack pointed by the SP and stores it at E.
00BA	POP F	POPs the data stored in the stack pointed by the SP and stores it at F.
00BB	POP G	POPs two data stored in the stack pointed by the SP and stores them at G.
00BC	POP H	POPs two data stored in the stack pointed by the SP and stores them at H.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
00BD	POP I	POPs two data stored in the stack pointed by the SP and stores them at I.
00BE	POP FL	POPs the data stored in the stack pointed by the SP and stores it in the Flag register.
00BF	PUSH A	PUSHes the 16-bit data stored in the Accumulator and stores them in the stack pointed by the SP.
00C0	PUSH B	PUSHes the 16-bit data stored in B and stores them in the stack pointed by the SP.
00C1	PUSH C	PUSHes the 16-bit data stored in C and stores them in the stack pointed by the SP.
00C2	PUSH D	PUSHes the 16-bit data stored in D and stores them in the stack pointed by the SP.
00C3	PUSH E	PUSHes the 16-bit data stored in E and stores them in the stack pointed by the SP.
00C4	PUSH F	PUSHes the 16-bit data stored in F and stores them in the stack pointed by the SP.
00C5	PUSH G	PUSHes 32-bit data stored at G and stores them in the stack pointed by the SP.
00C6	PUSH H	PUSHes 32-bit data stored at H and stores them in the stack pointed by the SP.
00C7	PUSH I	PUSHes 32-bit data stored at I and stores them in the stack pointed by the SP.
00C8	PUSH FL	PUSHes the 16-bit Flag register and stores it in the stack pointed by the SP.
00C9	PUSM A	PUSHes the data stored at the memory location pointed by the A/B Register pair into the stack pointed by the SP.
00CA	PUSM C	PUSHes the data stored at the memory location pointed by the C/D Register pair into the stack pointed by the SP.
00CB	PUSM E	PUSHes the data stored at the memory location pointed by the E/F Register pair into the stack pointed by the SP.
00CC	PUSM G	PUSHes the data stored at the memory location pointed by G into the stack pointed by the SP.
00CD	PUSM H	PUSHes the data stored at the memory location pointed by H into the stack pointed by the SP.
0OCE	PUSM I	PUSHes the data stored at the memory location pointed by I into the stack pointed by the SP.
00CF	POPM A	POPs the data stored in the stack pointed by the SP into the memory location pointed by the A/B Register pair.
00D0	POPM C	POPs the data stored in the stack pointed by the SP into the memory location pointed by the C/D Register pair.
00D1	POPM E	POPs the data stored in the stack pointed by the SP pair into the memory location pointed by the E/F Register.
00D2	POPM G	POPs the data stored in the stack pointed by the SP into the memory location pointed by G.
00D3	POPM H	POPs the data stored in the stack pointed by the SP into the memory location pointed by H.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
00D4	POPM I	POPs the data stored in the stack pointed by the SP into the memory location pointed by I.
00D5	XCHG A,B	Exchanges the data stored at the accumulator and the B register.
00D6	XCHG A,C	Exchanges the data stored at the accumulator and the C register.
00D7	XCHG A,D	Exchanges the data stored at the accumulator and the D register.
00D8	XCHG A,E	Exchanges the data stored at the accumulator and the E register.
00D9	XCHG A,F	Exchanges the data stored at the accumulator and the F register.
00DA	XCHG A,GH	Exchanges the data stored at the accumulator and the 16 higher order bits of the G register.
00DB	XCHG A,GL	Exchanges the data stored at the accumulator and the 16 lower order bits of the G register.
00DC	XCHG A,HH	Exchanges the data stored at the accumulator and the 16 higher order bits of the H register.
00DD	XCHG A,HL	Exchanges the data stored at the accumulator and the 16 lower order bits of the H register.
00DE	XCHG A,IH	Exchanges the data stored at the accumulator and the 16 higher order bits of the I register.
00DF	XCHG A,IL	Exchanges the data stored at the accumulator and the 16 lower order bits of the I register.
00E0	XCHG B,C	Exchanges the data stored at the B register and the C register.
00E1	XCHG B,D	Exchanges the data stored at the B register and the D register.
00E2	XCHG B,E	Exchanges the data stored at the B register and the E register.
00E3	XCHG B,F	Exchanges the data stored at the B register and the F register.
00E4	XCHG B,GH	Exchanges the data stored at the B register and the 16 higher order bits of the G register.
00E5	XCHG B,GL	Exchanges the data stored at the B register and the 16 lower order bits of the G register.
00E6	XCHG B,HH	Exchanges the data stored at the B register and the 16 higher order bits of the H register.
00E7	XCHG B,HL	Exchanges the data stored at the B register and the 16 lower order bits of the H register.
00E8	XCHG B,IH	Exchanges the data stored at the B register and the 16 higher order bits of the I register.
00E9	XCHG B,IL	Exchanges the data stored at the B register and the 16 lower order bits of the I register.
00EA	XCHG C,D	Exchanges the data stored at the C register and the D register.
00EB	XCHG C,E	Exchanges the data stored at the C register and the E register.
00EC	XCHG C,F	Exchanges the data stored at the C register and the F register.
00ED	XCHG C,GH	Exchanges the data stored at the C register and the 16 higher order bits of the G register.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
00EE	XCHG C,GL	Exchanges the data stored at the C register and the 16 lower order bits of the G register.
00EF	XCHG C,HH	Exchanges the data stored at the C register and the 16 higher order bits of the H register.
00F0	XCHG C,HL	Exchanges the data stored at the C register and the 16 lower order bits of the H register.
00F1	XCHG C,IH	Exchanges the data stored at the C register and the 16 higher order bits of the I register.
00F2	XCHG C,IL	Exchanges the data stored at the C register and the 16 lower order bits of the I register.
00F3	XCHG D,E	Exchanges the data stored at the D register and the E register.
00F4	XCHG D,F	Exchanges the data stored at the D register and the F register.
00F5	XCHG D,GH	Exchanges the data stored at the D register and the 16 higher order bits of the G register.
00F6	XCHG D,GL	Exchanges the data stored at the D register and the 16 lower order bits of the G register.
00F7	XCHG D,HH	Exchanges the data stored at the D register and the 16 higher order bits of the H register.
00F8	XCHG D,HL	Exchanges the data stored at the D register and the 16 lower order bits of the H register.
00F9	XCHG D,IH	Exchanges the data stored at the D register and the 16 higher order bits of the I register.
00FA	XCHG D,IL	Exchanges the data stored at the D register and the 16 lower order bits of the I register.
00FB	XCHG E,F	Exchanges the data stored at the E register and the F register.
00FC	XCHG E,GH	Exchanges the data stored at the E register and the 16 higher order bits of the G register.
00FD	XCHG E,GL	Exchanges the data stored at the E register and the 16 lower order bits of the G register.
00FE	XCHG E,HH	Exchanges the data stored at the E register and the 16 higher order bits of the H register.
00FF	XCHG E,HL	Exchanges the data stored at the E register and the 16 lower order bits of the H register.
0100	XCHG E,IH	Exchanges the data stored at the E register and the 16 higher order bits of the I register.
0101	XCHG E,IL	Exchanges the data stored at the E register and the 16 lower order bits of the I register.
0102	XCHG F,GH	Exchanges the data stored at the F register and the 16 higher order bits of the G register.
0103	XCHG F,GL	Exchanges the data stored at the F register and the 16 lower order bits of the G register.
0104	XCHG F,HH	Exchanges the data stored at the F register and the 16 higher order bits of the H register.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
0105	XCHG F,HL	Exchanges the data stored at the F register and the 16 lower order bits of the H register.
0106	XCHG F,IH	Exchanges the data stored at the F register and the 16 higher order bits of the I register.
0107	XCHG F,IL	Exchanges the data stored at the F register and the 16 lower order bits of the I register.
0108	XCHG GH,GL	Exchanges the data stored at the 16 higher order bits of the G register and the 16 lower order bits of the G register.
0109	XCHG GH,HH	Exchanges the data stored at the 16 higher order bits of the G register and the 16 higher order bits of the H register.
010A	XCHG GH,HL	Exchanges the data stored at the 16 higher order bits of the G register and the 16 lower order bits of the H register.
010B	XCHG GH,IH	Exchanges the data stored at the 16 higher order bits of the G register and the 16 higher order bits of the I register.
010C	XCHG GH,IL	Exchanges the data stored at the 16 higher order bits of the G register and the 16 lower order bits of the I register.
010D	XCHG GL,HH	Exchanges the data stored at the 16 lower order bits of the G register and the 16 higher order bits of the H register.
010E	XCHG GL,HL	Exchanges the data stored at the 16 lower order bits of the G register and the 16 lower order bits of the H register.
010F	XCHG GL,IH	Exchanges the data stored at the 16 lower order bits of the G register and the 16 higher order bits of the I register.
0110	XCHG GL,IL	Exchanges the data stored at the 16 lower order bits of the G register and the 16 lower order bits of the I register.
0111	XCHG HH,HL	Exchanges the data stored at the 16 higher order bits of the H register and the 16 lower order bits of the H register.
0112	XCHG HH,IH	Exchanges the data stored at the 16 higher order bits of the H register and the 16 higher order bits of the I register.
0113	XCHG HH,IL	Exchanges the data stored at the 16 higher order bits of the H register and the 16 lower order bits of the I register.
0114	XCHG HL,IH	Exchanges the data stored at the 16 lower order bits of the H register and the 16 higher order bits of the I register.
0115	XCHG HL,IL	Exchanges the data stored at the 16 lower order bits of the H register and the 16 lower order bits of the I register.
0116	XCHG IH,IL	Exchanges the data stored at the 16 higher order bits of the I register and the 16 lower order bits of the I register.
0117	XCHP A,C	Exchanges the data stored in the A/B register pair with that stored in the C/D register pair.
0118	XCHP A, E	Exchanges the data stored in the A/B register pair with that stored in the E/F register pair.
0119	XCHP A,G	Exchanges the data stored in the A/B register pair with that stored in the G register.
011A	XCHP A,H	Exchanges the data stored in the A/B register pair with that stored in the H register.
011B	XCHP A,I	Exchanges the data stored in the A/B register pair with that stored in the I register.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
011C	XCHP C, E	Exchanges the data stored in the C/D register pair with that stored in the E/F register pair.
011D	XCHP C,G	Exchanges the data stored in the C/D register pair with that stored in the G register.
011E	XCHP C,H	Exchanges the data stored in the C/D register pair with that stored in the H register.
011F	XCHP C,I	Exchanges the data stored in the C/D register pair with that stored in the I register.
0120	XCHP E,G	Exchanges the data stored in the E/F register pair with that stored in the G register.
0121	XCHP E,H	Exchanges the data stored in the E/F register pair with that stored in the H register.
0122	XCHP E,I	Exchanges the data stored in the E/F register pair with that stored in the I register.
0123	XCHP G,H	Exchanges the data stored in the G register with that stored in the H register.
0124	XCHP G,I	Exchanges the data stored in the G register with that stored in the I register.
0125	XCHP H,I	Exchanges the data stored in the H register with that stored in the I register.
0126	XCHP I,SP	Exchanges the data stored in the I register with that stored in the SP.
0127	STA address	Stores the data of the accumulator in the memory location given.
0128	STAX C	Stores the data of the accumulator in the memory location pointed by the value in the C/D register pair.
0129	STAX E	Stores the data of the accumulator in the memory location pointed by the value in the E/F register pair.
012A	STAX G	Stores the data of the accumulator in the memory location pointed by the value in the G register.
012B	STAX H	Stores the data of the accumulator in the memory location pointed by the value in the H register.
0200	EI	Enables all Interrupts.
0201	DI	Disables all Interrupts.
0202	HLT	The MPU executes the current instruction and enters the HALT phase. A reset or interrupt is necessary for it to exit the HALT phase.
0203	CLI	Resets all interrupts and sets the EI flip-flop.
0204	STI data	Enables selective interrupts by copying the data to the Interrupt-Mask register.
0205	RDI	Reads the status of the Interrupt-Mask register by copying its contents to the accumulator.
0206	LOCK	Enables the LOCK OUT signal pin, thus preventing all other devices from accessing the system bus. Interrupts are however not affected.
0207	DLCK	This instruction disables the LOCK OUT signal pin, and so it must be preceded be a LOCK command. Interrupts are however not affected.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
0208	CLF	Clears all flags.
0209	CLC	Clears the Carry Flag.
020A	STC	Sets the Carry Flag.
020B	STF data	Sets selective flags by copying the data to the flag register.
020C	RST 0	It is the software interrupt RST 0, it will transfer the execution of the microprocessor to the address (00000000)H
020D	RST 1	It is the software interrupt RST 1, it will transfer the execution of the microprocessor to the address (00000010)H
020E	RST 2	It is the software interrupt RST 2, it will transfer the execution of the microprocessor to the address (00000020)H
020F	RST 3	It is the software interrupt RST 3, it will transfer the execution of the microprocessor to the address (00000030)H
0210	RST 4	It is the software interrupt RST 4, it will transfer the execution of the microprocessor to the address (00000040)H
0211	RST 5	It is the software interrupt RST 5, it will transfer the execution of the microprocessor to the address (00000050)H
0212	RST 6	It is the software interrupt RST 6, it will transfer the execution of the microprocessor to the address (00000060)H
0213	RST 7	It is the software interrupt RST 7, it will transfer the execution of the microprocessor to the address (00000070)H
0214	RST 8	It is the software interrupt RST 8, it will transfer the execution of the microprocessor to the address (00000080)H
0215	RST 9	It is the software interrupt RST 9, it will transfer the execution of the microprocessor to the address (00000090)H
0216	RST 10	It is the software interrupt RST 10, it will transfer the execution of the microprocessor to the address (000000A0)H
0217	RST 11	It is the software interrupt RST 11, it will transfer the execution of the microprocessor to the address (000000B0)H
0218	RST 12	It is the software interrupt RST 12, it will transfer the execution of the microprocessor to the address (000000C0)H
0219	RST 13	It is the software interrupt RST 13, it will transfer the execution of the microprocessor to the address (000000D0)H
021A	RST 14	It is the software interrupt RST 14, it will transfer the execution of the microprocessor to the address (000000E0)H
021B	RST 15	It is the software interrupt RST 15, it will transfer the execution of the microprocessor to the address (000000F0)H
0300	ADD B	Adds the values stored at accumulator and B register and stores the result in accumulator.
0301	ADD C	Adds the values stored at accumulator and C register and stores the result in accumulator.
0302	ADD D	Adds the values stored at accumulator and D register and stores the result in accumulator.
0303	ADD E	Adds the values stored at accumulator and E register and stores the result in accumulator.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
0304	ADD F	Adds the values stored at accumulator and F register and stores the result in accumulator.
0305	ADD GH	Adds the values stored at accumulator and 16 higher order bits of G register and stores the result in accumulator.
0306	ADD GL	Adds the values stored at accumulator and 16 lower order bits of G register and stores the result in accumulator.
0307	ADD HH	Adds the values stored at accumulator and 16 higher order bits of H register and stores the result in accumulator.
0308	ADD HL	Adds the values stored at accumulator and 16 lower order bits of H register and stores the result in accumulator.
0309	ADD IH	Adds the values stored at accumulator and 16 higher order bits of I register and stores the result in accumulator.
030A	ADD IL	Adds the values stored at accumulator and 16 lower order bits of I register and stores the result in accumulator.
030B	ADD M	Adds the values stored at accumulator and memory and stores the result in accumulator.
030C	ADC B	Adds the values of the accumulator, the B register and the Carry flag and stores the result in the accumulator. The previous Carry flag is then reset.
030D	ADC C	Adds the values of the accumulator, the C register and the Carry flag and stores the result in the accumulator. The previous Carry flag is then reset.
030E	ADC D	Adds the values of the accumulator, the D register and the Carry flag and stores the result in the accumulator. The previous Carry flag is then reset.
030F	ADC E	Adds the values of the accumulator, the E register and the Carry flag and stores the result in the accumulator. The previous Carry flag is then reset.
0310	ADC F	Adds the values of the accumulator, the F register and the Carry flag and stores the result in the accumulator. The previous Carry flag is then reset.
0311	ADC GH	Adds the values of the accumulator, 16 higher order bits of G register and the Carry flag and stores the result in the accumulator. The previous Carry flag is then reset.
0312	ADC GL	Adds the values of the accumulator, 16 lower order bits of G register and the Carry flag and stores the result in the accumulator. The previous Carry flag is then reset.
0313	ADC HH	Adds the values of the accumulator, 16 higher order bits of H register and the Carry flag and stores the result in the accumulator. The previous Carry flag is then reset.
0314	ADC HL	Adds the values of the accumulator, 16 lower order bits of H register and the Carry flag and stores the result in the accumulator. The previous Carry flag is then reset.
0315	ADC IH	Adds the values of the accumulator, 16 higher order bits of I register and the Carry flag and stores the result in the accumulator. The previous Carry flag is then reset.
0316	ADC IL	Adds the values of the accumulator, 16 lower order bits of I register and the Carry flag and stores the result in the accumulator. The previous Carry flag is then reset.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
0317	ADC M	Adds the values of the accumulator, the memory and the Carry flag and stores the result in the accumulator. The previous Carry flag is then reset.
0318	ADI data	Adds the value of the data to that of the accumulator and stores the result in the accumulator.
0319	ACI data	Adds the value of the data to that of the accumulator along with the previous carry and stores the result in the accumulator. The previous Carry flag is reset.
031A	ADM address	Adds the value of the data stored at the memory location pointed by the address to the accumulator and stores the result in the accumulator.
031B	ACM address	Adds the value of the data stored at the memory location pointed by the address to the accumulator along with the previous carry and stores the result in the accumulator. The previous Carry flag is reset.
031C	SUB B	Subtracts the value stored at B register from that stored in the accumulator and stores the result in the accumulator.
031D	SUB C	Subtracts the value stored at C register from that stored in the accumulator and stores the result in the accumulator.
031E	SUB D	Subtracts the value stored at D register from that stored in the accumulator and stores the result in the accumulator.
031F	SUB E	Subtracts the value stored at E register from that stored in the accumulator and stores the result in the accumulator.
0320	SUB F	Subtracts the value stored at F register from that stored in the accumulator and stores the result in the accumulator.
0321	SUB GH	Subtracts the value stored in the 16 higher order bits of G register from that stored in the accumulator and stores the result in the accumulator.
0322	SUB GL	Subtracts the value stored in the 16 lower order bits of G register from that stored in the accumulator and stores the result in the accumulator.
0323	SUB HH	Subtracts the value stored in the 16 higher order bits of H register from that stored in the accumulator and stores the result in the accumulator.
0324	SUB HL	Subtracts the value stored in the 16 lower order bits of H register from that stored in the accumulator and stores the result in the accumulator.
0325	SUB IH	Subtracts the value stored in the 16 higher order bits of I register from that stored in the accumulator and stores the result in the accumulator.
0326	SUB IL	Subtracts the value stored in the 16 lower order bits of I register from that stored in the accumulator and stores the result in the accumulator.
0327	SUB M	Subtracts the value stored at memory from that stored in the accumulator and stores the result in the accumulator.
0328	SBB B	Subtracts the value stored at the B register and the previous borrow from the value stored in the accumulator and stores the result in the accumulator.
0329	SBB C	Subtracts the value stored at the C register and the previous borrow from the value stored in the accumulator and stores the result in the accumulator.
032A	SBB D	Subtracts the value stored at the D register and the previous borrow from the value stored in the accumulator and stores the result in the accumulator.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
032B	SBB E	Subtracts the value stored at the E register and the previous borrow from the value stored in the accumulator and stores the result in the accumulator.
032C	SBB F	Subtracts the value stored at the F register and the previous borrow from the value stored in the accumulator and stores the result in the accumulator.
032D	SBB GH	Subtracts the value stored at the 16 higher order bits of G register and the previous borrow from the value stored in the accumulator and stores the result in the accumulator. The previous Borrow bit is reset.
032E	SBB GL	Subtracts the value stored at the 16 lower order bits of G register and the previous borrow from the value stored in the accumulator and stores the result in the accumulator. The previous Borrow bit is reset.
032F	SBB HH	Subtracts the value stored at the 16 higher order bits of H register and the previous borrow from the value stored in the accumulator and stores the result in the accumulator. The previous Borrow bit is reset.
0330	SBB HL	Subtracts the value stored at the 16 lower order bits of H register and the previous borrow from the value stored in the accumulator and stores the result in the accumulator. The previous Borrow bit is reset.
0331	SBB IH	Subtracts the value stored at the 16 higher order bits of I register and the previous borrow from the value stored in the accumulator and stores the result in the accumulator. The previous Borrow bit is reset.
0332	SBB IL	Subtracts the value stored at the 16 lower order bits of I register and the previous borrow from the value stored in the accumulator and stores the result in the accumulator. The previous Borrow bit is reset.
0333	SBB M	Subtracts the value stored at the memory and the previous borrow from the value stored in the accumulator and stores the result in the accumulator. The previous Borrow bit is reset.
0334	SUI data	Subtracts the value of the data from the value stored in the accumulator and stores the result in the accumulator.
0335	SBI data	Subtracts the value of the data and the previous borrow from the value stored in the accumulator and stores the result in the accumulator. The previous Borrow bit is reset.
0336	SMU address	Subtracts the value of the data stored at the address from the value stored in the accumulator and stores the result in the accumulator.
0337	SBM address	Subtracts the value of the data stored at the address and the previous borrow from the value stored in the accumulator and stores the result in the accumulator. The previous Borrow bit is reset.
0338	SUBA B	Subtracts the value stored in the accumulator from that stored at B register and stores the result in the accumulator.
0339	SUBA C	Subtracts the value stored in the accumulator from that stored at C register and stores the result in the accumulator.
033A	SUBA D	Subtracts the value stored in the accumulator from that stored at D register and stores the result in the accumulator.
033B	SUBA E	Subtracts the value stored in the accumulator from that stored at E register and stores the result in the accumulator.
033C	SUBA F	Subtracts the value stored in the accumulator from that stored at F register and stores the result in the accumulator.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
033D	SUBA GH	Subtracts the value stored in the accumulator from that stored in the 16 higher order bits of G register and stores the result in the accumulator.
033E	SUBA GL	Subtracts the value stored in the accumulator from that stored in the 16 lower order bits of G register and stores the result in the accumulator.
033F	SUBA HH	Subtracts the value stored in the accumulator from that stored in the 16 higher order bits of H register and stores the result in the accumulator.
0340	SUBA HL	Subtracts the value stored in the accumulator from that stored in the 16 lower order bits of H register and stores the result in the accumulator.
0341	SUBA IH	Subtracts the value stored in the accumulator from that stored in the 16 higher order bits of I register and stores the result in the accumulator.
0342	SUBA IL	Subtracts the value stored in the accumulator from that stored in the 16 lower order bits of I register and stores the result in the accumulator.
0343	SUBA M	Subtracts the value stored in the accumulator from that stored at memory and stores the result in the accumulator.
0344	SBBA B	Subtracts the value stored in the accumulator from the sum of the value stored at the B register and the previous borrow and stores the result in the accumulator. The previous borrow bit is reset.
0345	SBBA C	Subtracts the value stored in the accumulator from the sum of the value stored at the C register and the previous borrow and stores the result in the accumulator. The previous borrow bit is reset.
0346	SBBA D	Subtracts the value stored in the accumulator from the sum of the value stored at the D register and the previous borrow and stores the result in the accumulator. The previous borrow bit is reset.
0347	SBBA E	Subtracts the value stored in the accumulator from the sum of the value stored at the E register and the previous borrow and stores the result in the accumulator. The previous borrow bit is reset.
0348	SBBA F	Subtracts the value stored in the accumulator from the sum of the value stored at the F register and the previous borrow and stores the result in the accumulator. The previous borrow bit is reset.
0349	SBBA GH	Subtracts the value stored in the accumulator from the sum of the value stored at the 16 higher order bits of G register and the previous borrow and stores the result in the accumulator. The previous Borrow bit is reset.
034A	SBBA GL	Subtracts the value stored in the accumulator from the sum of the value stored at the 16 lower order bits of G register and the previous borrow and stores the result in the accumulator. The previous Borrow bit is reset.
034B	SBBA HH	Subtracts the value stored in the accumulator from the sum of the value stored at the 16 higher order bits of H register and the previous borrow and stores the result in the accumulator. The previous Borrow bit is reset.
034C	SBBA HL	Subtracts the value stored in the accumulator from the sum of the value stored at the 16 lower order bits of H register and the previous borrow and stores the result in the accumulator. The previous Borrow bit is reset.
034D	SBBA IH	Subtracts the value stored in the accumulator from the sum of the value stored at the 16 higher order bits of I register and the previous borrow and stores the result in the accumulator. The previous Borrow bit is reset.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
034E	SBBA IL	Subtracts the value stored in the accumulator from the sum of the value stored at the 16 lower order bits of I register and the previous borrow and stores the result in the accumulator. The previous Borrow bit is reset.
034F	SBBA M	Subtracts the value stored in the accumulator from the sum of the value stored at the memory and the previous borrow and stores the result in the accumulator. The previous Borrow bit is reset.
0350	SUIA data	Subtracts the value stored in the accumulator from the value of the data and stores the result in the accumulator.
0351	SBIA data	Subtracts the value stored in the accumulator from the sum of the value of the data and the previous borrow and stores the result in the accumulator. The previous Borrow bit is reset.
0352	SMUA address	Subtracts the value stored in the accumulator from the value of the data stored at the address and stores the result in the accumulator.
0353	SBMA address	Subtracts the value stored in the accumulator from the sum of the value of the data stored at the address and the previous borrow and stores the result in the accumulator. The previous Borrow bit is reset.
0355	MUL C	Multiplies the values stored at accumulator and C register and stores the result in accumulator.
0356	MUL D	Multiplies the values stored at accumulator and D register and stores the result in accumulator.
0357	MUL E	Multiplies the values stored at accumulator and E register and stores the result in accumulator.
0358	MUL F	Multiplies the values stored at accumulator and F register and stores the result in accumulator.
0359	MUL GH	Multiplies the values stored at accumulator and 16 higher order bits of G register and stores the result in accumulator.
035A	MUL GL	Multiplies the values stored at accumulator and 16 lower order bits of G register and stores the result in accumulator.
035B	MUL HH	Multiplies the values stored at accumulator and 16 higher order bits of H register and stores the result in accumulator.
035C	MUL HL	Multiplies the values stored at accumulator and 16 lower order bits of H register and stores the result in accumulator.
035D	MUL IH	Multiplies the values stored at accumulator and 16 higher order bits of I register and stores the result in accumulator.
035E	MUL IL	Multiplies the values stored at accumulator and 16 lower order bits of I register and stores the result in accumulator.
035F	MUL M	Multiplies the values stored at accumulator and memory and stores the result in accumulator.
0360	MUI data	Multiplies the value of the data to that of the accumulator and stores the result in the accumulator.
0361	MUM address	Multiplies the value of the data stored at the memory location pointed by the address to the accumulator and stores the result in the accumulator.
0363	DIV C	Divides the values stored at accumulator by C register and stores the result in accumulator.
0364	DIV D	Divides the values stored at accumulator by D register and stores the result in accumulator.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
0365	DIV E	Divides the values stored at accumulator by E register and stores the result in accumulator.
0366	DIV F	Divides the values stored at accumulator by F register and stores the result in accumulator.
0367	DIV GH	Divides the values stored at accumulator by 16 higher order bits of G register and stores the result in accumulator.
0368	DIV GL	Divides the values stored at accumulator by 16 lower order bits of G register and stores the result in accumulator.
0369	DIV HH	Divides the values stored at accumulator by 16 higher order bits of H register and stores the result in accumulator.
036A	DIV HL	Divides the values stored at accumulator by 16 lower order bits of H register and stores the result in accumulator.
036B	DIV IH	Divides the values stored at accumulator by 16 higher order bits of I register and stores the result in accumulator.
036C	DIV IL	Divides the values stored at accumulator by 16 lower order bits of I register and stores the result in accumulator.
036D	DIV M	Divides the values stored at accumulator by that stored at memory and stores the result in accumulator.
036E	DVI data	Divides the value of the accumulator by that of the data and stores the result in the accumulator.
036F	DVM address	Divides the value of the accumulator by the data stored at the memory location pointed by the address and stores the result in the accumulator.
0371	DIVA C	Divides the values stored at C register by accumulator and stores the result in accumulator.
0372	DIVA D	Divides the values stored at D register by accumulator and stores the result in accumulator.
0373	DIVA E	Divides the values stored at E register by accumulator and stores the result in accumulator.
0374	DIVA F	Divides the values stored at F register by accumulator and stores the result in accumulator.
0375	DIVA GH	Divides the values stored at 16 higher order bits of G register by accumulator and stores the result in accumulator.
0376	DIVA GL	Divides the values stored at 16 lower order bits of G register by accumulator and stores the result in accumulator.
0377	DIVA HH	Divides the values stored at 16 higher order bits of H register by accumulator and stores the result in accumulator.
0378	DIVA HL	Divides the values stored at 16 lower order bits of H register by accumulator and stores the result in accumulator.
0379	DIVA IH	Divides the values stored 16 higher order bits of I register at by accumulator and stores the result in accumulator.
037A	DIVA IL	Divides the values stored at 16 lower order bits of I register by accumulator and stores the result in accumulator.
037B	DIVA M	Divides the values stored at memory by that stored at accumulator and stores the result in accumulator.
037C	DVIA data	Divides the value of the data by that of the accumulator and stores the result in the accumulator.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
037D	DVMA address	Divides the value of the the accumulator by data stored at the memory location pointed by the address and stores the result in the accumulator.
037E	INR B	Increments the value of B register by 1. Flags are affected.
037F	INR C	Increments the value of C register by 1. Flags are affected.
0380	INR D	Increments the value of D register by 1. Flags are affected.
0381	INR E	Increments the value of E register by 1. Flags are affected.
0382	INR F	Increments the value of F register by 1. Flags are affected.
0383	INR GH	Increments the value of the 16 higher order bits of G register by 1. Flags are affected.
0384	INR GL	Increments the value of 16 lower order bits of G register by 1. Flags are affected.
0385	INR HH	Increments the value of the 16 higher order bits of H register by 1. Flags are affected.
0386	INR HL	Increments the value of 16 lower order bits of H register by 1. Flags are affected.
0387	INR IH	Increments the value of the 16 higher order bits of I register by 1. Flags are affected.
0388	INR IL	Increments the value of 16 lower order bits of I register by 1. Flags are affected.
0389	INR M	Increments the value of memory by 1. Flags are affected.
038A	INX C	Increments the value of C/D register pair by 1. Flags are not affected.
038B	INX E	Increments the value of E/F register pair by 1. Flags are not affected.
038C	INX G	Increments the value of G register by 1. Flags are not affected.
038D	INX H	Increments the value of H register by 1. Flags are not affected.
038E	INX I	Increments the value of I register by 1. Flags are not affected.
038F	INX SP	Increments the value of SP by 1. Flags are not affected.
0390	DCR B	Decrements the value of B register by 1. Flags are affected.
0391	DCR C	Decrements the value of C register by 1. Flags are affected.
0392	DCR D	Decrements the value of D register by 1. Flags are affected.
0393	DCR E	Decrements the value of E register by 1. Flags are affected.
0394	DCR F	Decrements the value of F register by 1. Flags are affected.
0395	DCR GH	Decrements the value of 16 higher order bits of G register by 1. Flags are affected.
0396	DCR GL	Decrements the value of 16 lower order bits of G register by 1. Flags are affected.
0397	DCR HH	Decrements the value of 16 higher order bits of H register by 1. Flags are affected.
0398	DCR HL	Decrements the value of 16 lower order bits of H register by 1. Flags are affected.
0399	DCR IH	Decrements the value of 16 higher order bits of I register by 1. Flags are affected.
039A	DCR IL	Decrements the value of 16 lower order bits of I register by 1. Flags are affected.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
039B	DCR M	Decrements the value of memory by 1. Flags are affected.
039C	DCX C	Decrements the value of C/D register pair by 1. Flags are not affected.
039D	DCX E	Decrements the value of E/F register pair by 1. Flags are not affected.
039E	DCX G	Decrements the value of G register by 1. Flags are not affected.
039F	DCX H	Decrements the value of H register by 1. Flags are not affected.
03A0	DCX I	Decrements the value of I register by 1. Flags are not affected.
03A1	DCX SP	Decrements the value of SP by 1. Flags are not affected.
03A2	ANA B	Performs the logical AND of the values stored in accumulator and B register and stores it in the accumulator.
03A3	ANA C	Performs the logical AND of the values stored in accumulator and C register and stores it in the accumulator.
03A4	ANA D	Performs the logical AND of the values stored in accumulator and D register and stores it in the accumulator.
03A5	ANA E	Performs the logical AND of the values stored in accumulator and E register and stores it in the accumulator.
03A6	ANA F	Performs the logical AND of the values stored in accumulator and 16 higher order bits of G register and stores it in the accumulator.
03A7	ANA GH	Performs the logical AND of the values stored in accumulator and 16 higher order bits of G register and stores it in the accumulator.
03A8	ANA GL	Performs the logical AND of the values stored in accumulator and 16 lower order bits of G register and stores it in the accumulator.
03A9	ANA HH	Performs the logical AND of the values stored in accumulator and 16 higher order bits of H register and stores it in the accumulator.
03AA	ANA HL	Performs the logical AND of the values stored in accumulator and 16 lower order bits of H register and stores it in the accumulator.
03AB	ANA IH	Performs the logical AND of the values stored in accumulator and 16 higher order bits of I register and stores it in the accumulator.
03AC	ANA IL	Performs the logical AND of the values stored in accumulator and 16 lower order bits of I register and stores it in the accumulator.
03AD	ANA M	Performs the logical AND of the values stored in accumulator and memory and stores it in the accumulator.
03AE	ANR B,C	Performs the logical AND of the values stored in B and C registers and stores the result in the accumulator.
03AF	ANR B,D	Performs the logical AND of the values stored in B and D registers and stores the result in the accumulator.
03B0	ANR B,E	Performs the logical AND of the values stored in B and E registers and stores the result in the accumulator.
03B1	ANR B,F	Performs the logical AND of the values stored in B and F registers and stores the result in the accumulator.
03B2	ANR B,GH	Performs the logical AND of the values stored in B and 16 higher order bits of G registers and stores the result in the accumulator.
03B3	ANR B,GL	Performs the logical AND of the values stored in B and 16 lower order bits of G registers and stores the result in the accumulator.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
03B4	ANR B,HH	Performs the logical AND of the values stored in B and 16 higher order bits of H registers and stores the result in the accumulator.
03B5	ANR B,HL	Performs the logical AND of the values stored in B and 16 lower order bits of H registers and stores the result in the accumulator.
03B6	ANR B,IH	Performs the logical AND of the values stored in B and 16 higher order bits of I registers and stores the result in the accumulator.
03B7	ANR B,IL	Performs the logical AND of the values stored in B and 16 lower order bits of I registers and stores the result in the accumulator.
03B8	ANR B,M	Performs the logical AND of the values stored in B register and memory and stores the result in the accumulator.
03B9	ANR C,D	Performs the logical AND of the values stored in C and D registers and stores the result in the accumulator.
03BA	ANR C,E	Performs the logical AND of the values stored in C and E registers and stores the result in the accumulator.
03BB	ANR C,F	Performs the logical AND of the values stored in C and F registers and stores the result in the accumulator.
03BC	ANR C,GH	Performs the logical AND of the values stored in C and 16 higher order bits of G registers and stores the result in the accumulator.
03BD	ANR C,GL	Performs the logical AND of the values stored in C and 16 lower order bits of G registers and stores the result in the accumulator.
03BE	ANR C,HH	Performs the logical AND of the values stored in C and 16 higher order bits of H registers and stores the result in the accumulator.
03BF	ANR C,HL	Performs the logical AND of the values stored in C and 16 lower order bits of H registers and stores the result in the accumulator.
03C0	ANR C,IH	Performs the logical AND of the values stored in C and 16 higher order bits of I registers and stores the result in the accumulator.
03C1	ANR C,IL	Performs the logical AND of the values stored in C and 16 lower order bits of I registers and stores the result in the accumulator.
03C2	ANR C,M	Performs the logical AND of the values stored in C register and memory and stores the result in the accumulator.
03C3	ANR D,E	Performs the logical AND of the values stored in D and E registers and stores the result in the accumulator.
03C4	ANR D,F	Performs the logical AND of the values stored in D and F registers and stores the result in the accumulator.
03C5	ANR D,GH	Performs the logical AND of the values stored in D and 16 higher order bits of G registers and stores the result in the accumulator.
03C6	ANR D,GL	Performs the logical AND of the values stored in D and 16 lower order bits of G registers and stores the result in the accumulator.
03C7	ANR D,HH	Performs the logical AND of the values stored in D and 16 higher order bits of H registers and stores the result in the accumulator.
03C8	ANR D,HL	Performs the logical AND of the values stored in D and 16 lower order bits of H registers and stores the result in the accumulator.
03C9	ANR D,IH	Performs the logical AND of the values stored in D and 16 higher order bits of I registers and stores the result in the accumulator.

Opcode	Mnemonics	Comments
03CA	ANR D,IL	Performs the logical AND of the values stored in D and 16 lower order bits of I registers and stores the result in the accumulator.
03CB	ANR D,M	Performs the logical AND of the values stored in D register and memory and stores the result in the accumulator.
03CC	ANR E,F	Performs the logical AND of the values stored in E and F registers and stores the result in the accumulator.
03CD	ANR E,GH	Performs the logical AND of the values stored in E and 16 higher order bits of G registers and stores the result in the accumulator.
03CE	ANR E,GL	Performs the logical AND of the values stored in E and 16 lower order bits of G registers and stores the result in the accumulator.
03CF	ANR E,HH	Performs the logical AND of the values stored in E and 16 higher order bits of H registers and stores the result in the accumulator.
03D0	ANR E,HL	Performs the logical AND of the values stored in E and 16 lower order bits of H registers and stores the result in the accumulator.
03D1	ANR E,IH	Performs the logical AND of the values stored in E and 16 higher order bits of I registers and stores the result in the accumulator.
03D2	ANR E,IL	Performs the logical AND of the values stored in E and 16 lower order bits of I registers and stores the result in the accumulator.
03D3	ANR E,M	Performs the logical AND of the values stored in E register and memory and stores the result in the accumulator.
03D4	ANR F,GH	Performs the logical AND of the values stored in E and 16 higher order bits of G registers and stores the result in the accumulator.
03D5	ANR F,GL	Performs the logical AND of the values stored in E and 16 lower order bits of G registers and stores the result in the accumulator.
03D6	ANR F,HH	Performs the logical AND of the values stored in E and 16 higher order bits of H registers and stores the result in the accumulator.
03D7	ANR F,HL	Performs the logical AND of the values stored in E and 16 lower order bits of H registers and stores the result in the accumulator.
03D8	ANR F,IH	Performs the logical AND of the values stored in E and 16 higher order bits of I registers and stores the result in the accumulator.
03D9	ANR F,IL	Performs the logical AND of the values stored in E and 16 lower order bits of I registers and stores the result in the accumulator.
03DA	ANR F,M	Performs the logical AND of the values stored in E register and memory and stores the result in the accumulator.
03DB	ANR GH,GL	Performs the logical AND of the values stored in 16 higher order bits of G and 16 lower order bits of G registers and stores the result in the accumulator.
03DC	ANR GH,HH	Performs the logical AND of the values stored in 16 higher order bits of G and 16 higher order bits of H registers and stores the result in the accumulator.
03DD	ANR GH,HL	Performs the logical AND of the values stored in 16 higher order bits of G and 16 lower order bits of H registers and stores the result in the accumulator.
03DE	ANR GH,IH	Performs the logical AND of the values stored in 16 higher order bits of G and 16 higher order bits of I registers and stores the result in the accumulator.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
03DF	ANR GH,IL	Performs the logical AND of the values stored in 16 higher order bits of G and 16 lower order bits of I registers and stores the result in the accumulator.
03E0	ANR GH,M	Performs the logical AND of the values stored in 16 higher order bits of G register and memory and stores the result in the accumulator.
03E1	ANR GL,HH	Performs the logical AND of the values stored in 16 lower order bits of G and 16 higher order bits of H registers and stores the result in the accumulator.
03E2	ANR GL,HL	Performs the logical AND of the values stored in 16 lower order bits of G and 16 lower order bits of H registers and stores the result in the accumulator.
03E3	ANR GL,IH	Performs the logical AND of the values stored in 16 lower order bits of G and 16 higher order bits of I registers and stores the result in the accumulator.
03E4	ANR GL,IL	Performs the logical AND of the values stored in 16 lower order bits of G and 16 lower order bits of I registers and stores the result in the accumulator.
03E5	ANR GL,M	Performs the logical AND of the values stored in 16 lower order bits of G register and memory and stores the result in the accumulator.
03E6	ANR HH,HL	Performs the logical AND of the values stored in 16 higher order bits of H and 16 lower order bits of H registers and stores the result in the accumulator.
03E7	ANR HH,IH	Performs the logical AND of the values stored in 16 higher order bits of H and 16 higher order bits of I registers and stores the result in the accumulator.
03E8	ANR HH,IL	Performs the logical AND of the values stored in 16 higher order bits of H and 16 lower order bits of I registers and stores the result in the accumulator.
03E9	ANR HH,M	Performs the logical AND of the values stored in 16 higher order bits of H register and memory and stores the result in the accumulator.
03EA	ANR HL,IH	Performs the logical AND of the values stored in 16 lower order bits of H and 16 higher order bits of I registers and stores the result in the accumulator.
03EB	ANR HL,IL	Performs the logical AND of the values stored in 16 lower order bits of H and 16 lower order bits of I registers and stores the result in the accumulator.
03EC	ANR HL,M	Performs the logical AND of the values stored in 16 lower order bits of H register and memory and stores the result in the accumulator.
03ED	ANR IH,IL	Performs the logical AND of the values stored in 16 higher order bits of I and 16 lower order bits of I registers and stores the result in the accumulator.
03EE	ANR IH,M	Performs the logical AND of the values stored in 16 higher order bits of I register and memory and stores the result in the accumulator.
03EF	ANR IL,M	Performs the logical AND of the values stored in 16 lower order bits of I register and memory and stores the result in the accumulator.
03F0	ANI data	Performs the logical AND of the data and the accumulator contents and stores the result in the accumulator.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
03F1	ANM address	Performs the logical AND of the data at the memory location pointed by the address and the contents of the accumulator and stores the result in the accumulator.
03F2	ORA B	Performs the logical OR of the values stored in accumulator and B register and stores it in the accumulator.
03F3	ORA C	Performs the logical OR of the values stored in accumulator and C register and stores it in the accumulator.
03F4	ORA D	Performs the logical OR of the values stored in accumulator and D register and stores it in the accumulator.
03F5	ORA E	Performs the logical OR of the values stored in accumulator and E register and stores it in the accumulator.
03F6	ORA F	Performs the logical OR of the values stored in accumulator and 16 higher order bits of G register and stores it in the accumulator.
03F7	ORA GH	Performs the logical OR of the values stored in accumulator and 16 higher order bits of G register and stores it in the accumulator.
03F8	ORA GL	Performs the logical OR of the values stored in accumulator and 16 lower order bits of G register and stores it in the accumulator.
03F9	ORA HH	Performs the logical OR of the values stored in accumulator and 16 higher order bits of H register and stores it in the accumulator.
03FA	ORA HL	Performs the logical OR of the values stored in accumulator and 16 lower order bits of H register and stores it in the accumulator.
03FB	ORA IH	Performs the logical OR of the values stored in accumulator and 16 higher order bits of I register and stores it in the accumulator.
03FC	ORA IL	Performs the logical OR of the values stored in accumulator and 16 lower order bits of I register and stores it in the accumulator.
03FD	ORA M	Performs the logical OR of the values stored in accumulator and memory and stores it in the accumulator.
03FE	ORR B,C	Performs the logical OR of the values stored in B and C registers and stores the result in the accumulator.
03FF	ORR B,D	Performs the logical OR of the values stored in B and D registers and stores the result in the accumulator.
0400	ORR B,E	Performs the logical OR of the values stored in B and E registers and stores the result in the accumulator.
0401	ORR B,F	Performs the logical OR of the values stored in B and F registers and stores the result in the accumulator.
0402	ORR B,GH	Performs the logical OR of the values stored in B and 16 higher order bits of G registers and stores the result in the accumulator.
0403	ORR B,GL	Performs the logical OR of the values stored in B and 16 lower order bits of G registers and stores the result in the accumulator.
0404	ORR B,HH	Performs the logical OR of the values stored in B and 16 higher order bits of H registers and stores the result in the accumulator.
0405	ORR B,HL	Performs the logical OR of the values stored in B and 16 lower order bits of H registers and stores the result in the accumulator.
0406	ORR B,IH	Performs the logical OR of the values stored in B and 16 higher order bits of I registers and stores the result in the accumulator.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
0407	ORR B,IL	Performs the logical OR of the values stored in B and 16 lower order bits of I registers and stores the result in the accumulator.
0408	ORR B,M	Performs the logical OR of the values stored in B register and memory and stores the result in the accumulator.
0409	ORR C,D	Performs the logical OR of the values stored in C and D registers and stores the result in the accumulator.
040A	ORR C,E	Performs the logical OR of the values stored in C and E registers and stores the result in the accumulator.
040B	ORR C,F	Performs the logical OR of the values stored in C and F registers and stores the result in the accumulator.
040C	ORR C,GH	Performs the logical OR of the values stored in C and 16 higher order bits of G registers and stores the result in the accumulator.
040D	ORR C,GL	Performs the logical OR of the values stored in C and 16 lower order bits of G registers and stores the result in the accumulator.
040E	ORR C,HH	Performs the logical OR of the values stored in C and 16 higher order bits of H registers and stores the result in the accumulator.
040F	ORR C,HL	Performs the logical OR of the values stored in C and 16 lower order bits of H registers and stores the result in the accumulator.
0410	ORR C,IH	Performs the logical OR of the values stored in C and 16 higher order bits of I registers and stores the result in the accumulator.
0411	ORR C,IL	Performs the logical OR of the values stored in C and 16 lower order bits of I registers and stores the result in the accumulator.
0412	ORR C,M	Performs the logical OR of the values stored in C register and memory and stores the result in the accumulator.
0413	ORR D,E	Performs the logical OR of the values stored in D and E registers and stores the result in the accumulator.
0414	ORR D,F	Performs the logical OR of the values stored in D and F registers and stores the result in the accumulator.
0415	ORR D,GH	Performs the logical OR of the values stored in D and 16 higher order bits of G registers and stores the result in the accumulator.
0416	ORR D,GL	Performs the logical OR of the values stored in D and 16 lower order bits of G registers and stores the result in the accumulator.
0417	ORR D,HH	Performs the logical OR of the values stored in D and 16 higher order bits of H registers and stores the result in the accumulator.
0418	ORR D,HL	Performs the logical OR of the values stored in D and 16 lower order bits of H registers and stores the result in the accumulator.
0419	ORR D,IH	Performs the logical OR of the values stored in D and 16 higher order bits of I registers and stores the result in the accumulator.
041A	ORR D,IL	Performs the logical OR of the values stored in D and 16 lower order bits of I registers and stores the result in the accumulator.
041B	ORR D,M	Performs the logical OR of the values stored in D register and memory and stores the result in the accumulator.
041C	ORR E,F	Performs the logical OR of the values stored in E and F registers and stores the result in the accumulator.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
041D	ORR E,GH	Performs the logical OR of the values stored in E and 16 higher order bits of G registers and stores the result in the accumulator.
041E	ORR E,GL	Performs the logical OR of the values stored in E and 16 lower order bits of G registers and stores the result in the accumulator.
041F	ORR E,HH	Performs the logical OR of the values stored in E and 16 higher order bits of H registers and stores the result in the accumulator.
0420	ORR E,HL	Performs the logical OR of the values stored in E and 16 lower order bits of H registers and stores the result in the accumulator.
0421	ORR E,IH	Performs the logical OR of the values stored in E and 16 higher order bits of I registers and stores the result in the accumulator.
0422	ORR E,IL	Performs the logical OR of the values stored in E and 16 lower order bits of I registers and stores the result in the accumulator.
0423	ORR E,M	Performs the logical OR of the values stored in E register and memory and stores the result in the accumulator.
0424	ORR F,GH	Performs the logical OR of the values stored in E and 16 higher order bits of G registers and stores the result in the accumulator.
0425	ORR F,GL	Performs the logical OR of the values stored in E and 16 lower order bits of G registers and stores the result in the accumulator.
0426	ORR F,HH	Performs the logical OR of the values stored in E and 16 higher order bits of H registers and stores the result in the accumulator.
0427	ORR F,HL	Performs the logical OR of the values stored in E and 16 lower order bits of H registers and stores the result in the accumulator.
0428	ORR F,IH	Performs the logical OR of the values stored in E and 16 higher order bits of I registers and stores the result in the accumulator.
0429	ORR F,IL	Performs the logical OR of the values stored in E and 16 lower order bits of I registers and stores the result in the accumulator.
042A	ORR F,M	Performs the logical OR of the values stored in E register and memory and stores the result in the accumulator.
042B	ORR GH,GL	Performs the logical OR of the values stored in 16 higher order bits of G and 16 lower order bits of G registers and stores the result in the accumulator.
042C	ORR GH,HH	Performs the logical OR of the values stored in 16 higher order bits of G and 16 higher order bits of H registers and stores the result in the accumulator.
042D	ORR GH,HL	Performs the logical OR of the values stored in 16 higher order bits of G and 16 lower order bits of H registers and stores the result in the accumulator.
042E	ORR GH,IH	Performs the logical OR of the values stored in 16 higher order bits of G and 16 higher order bits of I registers and stores the result in the accumulator.
042F	ORR GH,IL	Performs the logical OR of the values stored in 16 higher order bits of G and 16 lower order bits of I registers and stores the result in the accumulator.
0430	ORR GH,M	Performs the logical OR of the values stored in 16 higher order bits of G register and memory and stores the result in the accumulator.
0431	ORR GL,HH	Performs the logical OR of the values stored in 16 lower order bits of G and 16 higher order bits of H registers and stores the result in the accumulator.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
0432	ORR GL,HL	Performs the logical OR of the values stored in 16 lower order bits of G and 16 lower order bits of H registers and stores the result in the accumulator.
0433	ORR GL,IH	Performs the logical OR of the values stored in 16 lower order bits of G and 16 higher order bits of I registers and stores the result in the accumulator.
0434	ORR GL,IL	Performs the logical OR of the values stored in 16 lower order bits of G and 16 lower order bits of I registers and stores the result in the accumulator.
0435	ORR GL,M	Performs the logical OR of the values stored in 16 lower order bits of G register and memory and stores the result in the accumulator.
0436	ORR HH,HL	Performs the logical OR of the values stored in 16 higher order bits of H and 16 lower order bits of H registers and stores the result in the accumulator.
0437	ORR HH,IH	Performs the logical OR of the values stored in 16 higher order bits of H and 16 higher order bits of I registers and stores the result in the accumulator.
0438	ORR HH,IL	Performs the logical OR of the values stored in 16 higher order bits of H and 16 lower order bits of I registers and stores the result in the accumulator.
0439	ORR HH,M	Performs the logical OR of the values stored in 16 higher order bits of H register and memory and stores the result in the accumulator.
043A	ORR HL,IH	Performs the logical OR of the values stored in 16 lower order bits of H and 16 higher order bits of I registers and stores the result in the accumulator.
043B	ORR HL,IL	Performs the logical OR of the values stored in 16 lower order bits of H and 16 lower order bits of I registers and stores the result in the accumulator.
043C	ORR HL,M	Performs the logical OR of the values stored in 16 lower order bits of H register and memory and stores the result in the accumulator.
043D	ORR IH,IL	Performs the logical OR of the values stored in 16 higher order bits of I and 16 lower order bits of I registers and stores the result in the accumulator.
043E	ORR IH,M	Performs the logical OR of the values stored in 16 higher order bits of I register and memory and stores the result in the accumulator.
043F	ORR IL,M	Performs the logical OR of the values stored in 16 lower order bits of I register and memory and stores the result in the accumulator.
0440	ORI data	Performs the logical OR of the data and the accumulator contents and stores the result in the accumulator.
0441	ORM address	Performs the logical OR of the data at the memory location pointed by the address and the contents of the accumulator and stores the result in the accumulator.
0442	XORA B	Performs the logical XOR of the values stored in accumulator and B register and stores it in the accumulator.
0443	XORA C	Performs the logical XOR of the values stored in accumulator and C register and stores it in the accumulator.
0444	XORA D	Performs the logical XOR of the values stored in accumulator and D register and stores it in the accumulator.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
0445	XORA E	Performs the logical XOR of the values stored in accumulator and E register and stores it in the accumulator.
0446	XORA F	Performs the logical XOR of the values stored in accumulator and 16 higher order bits of G register and stores it in the accumulator.
0447	XORA GH	Performs the logical XOR of the values stored in accumulator and 16 higher order bits of G register and stores it in the accumulator.
0448	XORA GL	Performs the logical XOR of the values stored in accumulator and 16 lower order bits of G register and stores it in the accumulator.
0447	XORA HH	Performs the logical XOR of the values stored in accumulator and 16 higher order bits of H register and stores it in the accumulator.
0448	XORA HL	Performs the logical XOR of the values stored in accumulator and 16 lower order bits of H register and stores it in the accumulator.
0449	XORA IH	Performs the logical XOR of the values stored in accumulator and 16 higher order bits of I register and stores it in the accumulator.
044A	XORA IL	Performs the logical XOR of the values stored in accumulator and 16 lower order bits of I register and stores it in the accumulator.
044B	XORA M	Performs the logical XOR of the values stored in accumulator and memory and stores it in the accumulator.
044C	XORR B,C	Performs the logical XOR of the values stored in B and C registers and stores the result in the accumulator.
044D	XORR B,D	Performs the logical XOR of the values stored in B and D registers and stores the result in the accumulator.
044E	XORR B,E	Performs the logical XOR of the values stored in B and E registers and stores the result in the accumulator.
044F	XORR B,F	Performs the logical XOR of the values stored in B and F registers and stores the result in the accumulator.
0450	XORR B,GH	Performs the logical XOR of the values stored in B and 16 higher order bits of G registers and stores the result in the accumulator.
0451	XORR B,GL	Performs the logical XOR of the values stored in B and 16 lower order bits of G registers and stores the result in the accumulator.
0452	XORR B,HH	Performs the logical XOR of the values stored in B and 16 higher order bits of H registers and stores the result in the accumulator.
0453	XORR B,HL	Performs the logical XOR of the values stored in B and 16 lower order bits of H registers and stores the result in the accumulator.
0454	XORR B,IH	Performs the logical XOR of the values stored in B and 16 higher order bits of I registers and stores the result in the accumulator.
0455	XORR B,IL	Performs the logical XOR of the values stored in B and 16 lower order bits of I registers and stores the result in the accumulator.
0456	XORR B,M	Performs the logical XOR of the values stored in B register and memory and stores the result in the accumulator.
0457	XORR C,D	Performs the logical XOR of the values stored in C and D registers and stores the result in the accumulator.
0458	XORR C,E	Performs the logical XOR of the values stored in C and E registers and stores the result in the accumulator.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
0459	XORR C,F	Performs the logical XOR of the values stored in C and F registers and stores the result in the accumulator.
045A	XORR C,GH	Performs the logical XOR of the values stored in C and 16 higher order bits of G registers and stores the result in the accumulator.
045B	XORR C,GL	Performs the logical XOR of the values stored in C and 16 lower order bits of G registers and stores the result in the accumulator.
045C	XORR C,HH	Performs the logical XOR of the values stored in C and 16 higher order bits of H registers and stores the result in the accumulator.
045D	XORR C,HL	Performs the logical XOR of the values stored in C and 16 lower order bits of H registers and stores the result in the accumulator.
045E	XORR C,IH	Performs the logical XOR of the values stored in C and 16 higher order bits of I registers and stores the result in the accumulator.
045F	XORR C,IL	Performs the logical XOR of the values stored in C and 16 lower order bits of I registers and stores the result in the accumulator.
0460	XORR C,M	Performs the logical XOR of the values stored in C register and memory and stores the result in the accumulator.
0461	XORR D,E	Performs the logical XOR of the values stored in D and E registers and stores the result in the accumulator.
0462	XORR D,F	Performs the logical XOR of the values stored in D and F registers and stores the result in the accumulator.
0463	XORR D,GH	Performs the logical XOR of the values stored in D and 16 higher order bits of G registers and stores the result in the accumulator.
0464	XORR D,GL	Performs the logical XOR of the values stored in D and 16 lower order bits of G registers and stores the result in the accumulator.
0465	XORR D,HH	Performs the logical XOR of the values stored in D and 16 higher order bits of H registers and stores the result in the accumulator.
0466	XORR D,HL	Performs the logical XOR of the values stored in D and 16 lower order bits of H registers and stores the result in the accumulator.
0467	XORR D,IH	Performs the logical XOR of the values stored in D and 16 higher order bits of I registers and stores the result in the accumulator.
0468	XORR D,IL	Performs the logical XOR of the values stored in D and 16 lower order bits of I registers and stores the result in the accumulator.
0469	XORR D,M	Performs the logical XOR of the values stored in D register and memory and stores the result in the accumulator.
046A	XORR E,F	Performs the logical XOR of the values stored in E and F registers and stores the result in the accumulator.
046B	XORR E,GH	Performs the logical XOR of the values stored in E and 16 higher order bits of G registers and stores the result in the accumulator.
046C	XORR E,GL	Performs the logical XOR of the values stored in E and 16 lower order bits of G registers and stores the result in the accumulator.
046D	XORR E,HH	Performs the logical XOR of the values stored in E and 16 higher order bits of H registers and stores the result in the accumulator.
046E	XORR E,HL	Performs the logical XOR of the values stored in E and 16 lower order bits of H registers and stores the result in the accumulator.
046F	XORR E,IH	Performs the logical XOR of the values stored in E and 16 higher order bits of I registers and stores the result in the accumulator.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
0470	XORR E,IL	Performs the logical XOR of the values stored in E and 16 lower order bits of I registers and stores the result in the accumulator.
0471	XORR E,M	Performs the logical XOR of the values stored in E register and memory and stores the result in the accumulator.
0472	XORR F,GH	Performs the logical XOR of the values stored in E and 16 higher order bits of G registers and stores the result in the accumulator.
0473	XORR F,GL	Performs the logical XOR of the values stored in E and 16 lower order bits of G registers and stores the result in the accumulator.
0474	XORR F,HH	Performs the logical XOR of the values stored in E and 16 higher order bits of H registers and stores the result in the accumulator.
0475	XORR F,HL	Performs the logical XOR of the values stored in E and 16 lower order bits of H registers and stores the result in the accumulator.
0476	XORR F,IH	Performs the logical XOR of the values stored in E and 16 higher order bits of I registers and stores the result in the accumulator.
0477	XORR F,IL	Performs the logical XOR of the values stored in E and 16 lower order bits of I registers and stores the result in the accumulator.
0478	XORR F,M	Performs the logical XOR of the values stored in E register and memory and stores the result in the accumulator.
0479	XORR GH,GL	Performs the logical XOR of the values stored in 16 higher order bits of G and 16 lower order bits of G registers and stores the result in the accumulator.
047A	XORR GH,HH	Performs the logical XOR of the values stored in 16 higher order bits of G and 16 higher order bits of H registers and stores the result in the accumulator.
047B	XORR GH,HL	Performs the logical XOR of the values stored in 16 higher order bits of G and 16 lower order bits of H registers and stores the result in the accumulator.
047C	XORR GH,IH	Performs the logical XOR of the values stored in 16 higher order bits of G and 16 higher order bits of I registers and stores the result in the accumulator.
047D	XORR GH,IL	Performs the logical XOR of the values stored in 16 higher order bits of G and 16 lower order bits of I registers and stores the result in the accumulator.
047E	XORR GH,M	Performs the logical XOR of the values stored in 16 higher order bits of G register and memory and stores the result in the accumulator.
047F	XORR GL,HH	Performs the logical XOR of the values stored in 16 lower order bits of G and 16 higher order bits of H registers and stores the result in the accumulator.
0480	XORR GL,HL	Performs the logical XOR of the values stored in 16 lower order bits of G and 16 lower order bits of H registers and stores the result in the accumulator.
0481	XORR GL,IH	Performs the logical XOR of the values stored in 16 lower order bits of G and 16 higher order bits of I registers and stores the result in the accumulator.
0482	XORR GL,IL	Performs the logical XOR of the values stored in 16 lower order bits of G and 16 lower order bits of I registers and stores the result in the accumulator.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
0483	XORR GL,M	Performs the logical XOR of the values stored in 16 lower order bits of G register and memory and stores the result in the accumulator.
0484	XORR HH,HL	Performs the logical XOR of the values stored in 16 higher order bits of H and 16 lower order bits of H registers and stores the result in the accumulator.
0485	XORR HH,IH	Performs the logical XOR of the values stored in 16 higher order bits of H and 16 higher order bits of I registers and stores the result in the accumulator.
0486	XORR HH,IL	Performs the logical XOR of the values stored in 16 higher order bits of H and 16 lower order bits of I registers and stores the result in the accumulator.
0487	XORR HH,M	Performs the logical XOR of the values stored in 16 higher order bits of H register and memory and stores the result in the accumulator.
0488	XORR HL,IH	Performs the logical XOR of the values stored in 16 lower order bits of H and 16 higher order bits of I registers and stores the result in the accumulator.
0489	XORR HL,IL	Performs the logical XOR of the values stored in 16 lower order bits of H and 16 lower order bits of I registers and stores the result in the accumulator.
048A	XORR HL,M	Performs the logical XOR of the values stored in 16 lower order bits of H register and memory and stores the result in the accumulator.
048B	XORR IH,IL	Performs the logical XOR of the values stored in 16 higher order bits of I and 16 lower order bits of I registers and stores the result in the accumulator.
048C	XORR IH,M	Performs the logical XOR of the values stored in 16 higher order bits of I register and memory and stores the result in the accumulator.
048D	XORR IL,M	Performs the logical XOR of the values stored in 16 lower order bits of I register and memory and stores the result in the accumulator.
048E	XORI data	Performs the logical XOR of the data and the accumulator contents and stores the result in the accumulator.
048F	XORM address	Performs the logical XOR of the data at the memory location pointed by the address and the contents of the accumulator and stores the result in the accumulator.
0490	NOT A	Performs the logical NOT of the value stored in accumulator and stores it back in the accumulator.
0491	NOT B	Performs the logical NOT of the value stored in B register and stores it back in the B register.
0492	NOT C	Performs the logical NOT of the value stored in C register and stores it back in the C register.
0493	NOT D	Performs the logical NOT of the value stored in D register and stores it back in the D register.
0494	NOT E	Performs the logical NOT of the value stored in E register and stores it back in the E register.
0495	NOT F	Performs the logical NOT of the value stored in F register and stores it back in the F register.
0496	NOT GH	Performs the logical NOT of the value stored in the 16 higher order bits of G register and stores it back in the 16 higher order bits of G register.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
0497	NOT GL	Performs the logical NOT of the value stored in the 16 lower order bits of G register and stores it back in the 16 lower order bits of G register.
0498	NOT HH	Performs the logical NOT of the value stored in the 16 higher order bits of H register and stores it back in the 16 higher order bits of H register.
0499	NOT HL	Performs the logical NOT of the value stored in the 16 lower order bits of H register and stores it back in the 16 lower order bits of H register.
049A	NOT IH	Performs the logical NOT of the value stored in the 16 higher order bits of I register and stores it back in the 16 higher order bits of I register.
049B	NOT IL	Performs the logical NOT of the value stored in the 16 lower order bits of I register and stores it back in the 16 lower order bits of I register.
049C	NOT M	Performs the logical NOT of the value stored in the memory and stores it back in the memory.
049D	NOA B	Performs the logical NOT of the value stored in B register and stores it in the accumulator.
049E	NOA C	Performs the logical NOT of the value stored in C register and stores it in the accumulator.
049F	NOA D	Performs the logical NOT of the value stored in D register and stores it in the accumulator.
04A0	NOA E	Performs the logical NOT of the value stored in E register and stores it in the accumulator.
04A1	NOA F	Performs the logical NOT of the value stored in F register and stores it in the accumulator.
04A2	NOA GH	Performs the logical NOT of the value stored in the 16 higher order bits of G register and stores it in the accumulator.
04A3	NOA GL	Performs the logical NOT of the value stored in the 16 lower order bits of G register and stores it in the accumulator.
04A4	NOA HH	Performs the logical NOT of the value stored in the 16 higher order bits of H register and stores it in the accumulator.
04A5	NOA HL	Performs the logical NOT of the value stored in the 16 lower order bits of H register and stores it in the accumulator.
04A6	NOA IH	Performs the logical NOT of the value stored in the 16 higher order bits of I register and stores it in the accumulator.
04A7	NOA IL	Performs the logical NOT of the value stored in the 16 lower order bits of I register and stores it in the accumulator.
04A8	NOA M	Performs the logical NOT of the value stored in the memory and stores it in the accumulator.
04A9	NOI data	Performs the logical NOT of the data and the contents of the accumulator and stores the result in the accumulator.
04AA	NOM address	Performs the logical NOT of the data at the memory location pointed by the address and stores the result in the accumulator.
04AB	RCR	Perform 1-bit right circular rotation of the contents of the accumulator through the carry bit.
04AC	RRC	Performs 1-bit right circular rotation of the contents of the accumulator. The higher order bit of the accumulator is copied to the carry and it shifts to the lower order bit of the accumulator.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
04AD	RRA	Performs 1-bit right circular rotation of the contents of the accumulator. The carry bit is not affected.
04AE	LCR	Perform 1-bit left circular rotation of the contents of the accumulator through the carry bit.
04AF	LRC	Performs 1-bit left circular rotation of the contents of the accumulator. The lower order bit of the accumulator is copied to the carry and it shifts to the higher order bit of the accumulator.
04B1	LRA	Performs 1-bit left circular rotation of the contents of the accumulator. The carry bit is not affected.
04B2	SHR	Perform 1-bit right shift of the contents of the accumulator.
04B3	SRC	Performs 1-bit right shft of the contents of the accumulator through the carry bit.
04B4	SHL	Perform 1-bit left shift of the contents of the accumulator.
04B5	SLC	Performs 1-bit left shift of the contents of the accumulator through the carry bit.
04B6	NEG	Inverts the sign of the value stored in the accumulator. For this to work, it assumes that the value stored is a signed number.
04B7	CPA B	Compares the value of the accumulator with that of the B register and sets the flags accordingly.
04B8	CPA C	Compares the value of the accumulator with that of the C register and sets the flags accordingly.
04B9	CPA D	Compares the value of the accumulator with that of the D register and sets the flags accordingly.
04BA	CPA E	Compares the value of the accumulator with that of the E register and sets the flags accordingly.
04BB	CPA F	Compares the value of the accumulator with that of the F register and sets the flags accordingly.
04BC	CPA GH	Compares the value of the accumulator with that of the 16 higher order bits of the G register and sets the flags accordingly.
04BD	CPA GL	Compares the value of the accumulator with that of the 16 lower order bits of the G register and sets the flags accordingly.
04BE	CPA HH	Compares the value of the accumulator with that of the 16 higher order bits of the H register and sets the flags accordingly.
04BF	CPA HL	Compares the value of the accumulator with that of the 16 lower order bits of the H register and sets the flags accordingly.
04C0	CPA IH	Compares the value of the accumulator with that of the 16 higher order bits of the I register and sets the flags accordingly.
04C1	CPA IL	Compares the value of the accumulator with that of the 16 lower order bits of the I register and sets the flags accordingly.
04C2	CPA M	Compares the value of the accumulator with that of the memory and sets the flags accordingly.
04C3	CMP B,C	Compares the value of the B register with that of the C register and sets the flags accordingly.
04C4	CMP B,D	Compares the value of the B register with that of the D register and sets the flags accordingly.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
04C5	CMP B,E	Compares the value of the B register with that of the E register and sets the flags accordingly.
04C6	CMP B,F	Compares the value of the B register with that of the F register and sets the flags accordingly.
04C7	CMP B,GH	Compares the value of the B register with that of the 16 higher order bits of the G register and sets the flags accordingly.
04C8	CMP B,GL	Compares the value of the B register with that of the 16 lower order bits of the G register and sets the flags accordingly.
04C9	CMP B,HH	Compares the value of the B register with that of the 16 higher order bits of the H register and sets the flags accordingly.
04CA	CMP B,HL	Compares the value of the B register with that of the 16 lower order bits of the H register and sets the flags accordingly.
04CB	CMP B,IH	Compares the value of the B register with that of the 16 higher order bits of the I register and sets the flags accordingly.
04CC	CMP B,IL	Compares the value of the B register with that of the 16 lower order bits of the I register and sets the flags accordingly.
04CD	CMP B,M	Compares the value of the B register with that of the memory and sets the flags accordingly.
04CE	CMP C,D	Compares the value of the C register with that of the D register and sets the flags accordingly.
04CF	CMP C,E	Compares the value of the C register with that of the E register and sets the flags accordingly.
04D0	CMP C,F	Compares the value of the C register with that of the F register and sets the flags accordingly.
04D1	CMP C,GH	Compares the value of the C register with that of the 16 higher order bits of the G register and sets the flags accordingly.
04D2	CMP C,GL	Compares the value of the C register with that of the 16 lower order bits of the G register and sets the flags accordingly.
04D3	CMP C,HH	Compares the value of the C register with that of the 16 higher order bits of the H register and sets the flags accordingly.
04D4	CMP C,HL	Compares the value of the C register with that of the 16 lower order bits of the H register and sets the flags accordingly.
04D5	CMP C,IH	Compares the value of the C register with that of the 16 higher order bits of the I register and sets the flags accordingly.
04D6	CMP C,IL	Compares the value of the C register with that of the 16 lower order bits of the I register and sets the flags accordingly.
04D7	CMP C,M	Compares the value of the C register with that of the memory and sets the flags accordingly.
04D8	CMP D,E	Compares the value of the D register with that of the E register and sets the flags accordingly.
04D9	CMP D,F	Compares the value of the D register with that of the F register and sets the flags accordingly.
04DA	CMP D,GH	Compares the value of the D register with that of the 16 higher order bits of the G register and sets the flags accordingly.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
04DB	CMP D,GL	Compares the value of the D register with that of the 16 lower order bits of the G register and sets the flags accordingly.
04DC	CMP D,HH	Compares the value of the D register with that of the 16 higher order bits of the H register and sets the flags accordingly.
04DD	CMP D,HL	Compares the value of the D register with that of the 16 lower order bits of the H register and sets the flags accordingly.
04DE	CMP D,IH	Compares the value of the D register with that of the 16 higher order bits of the I register and sets the flags accordingly.
04DF	CMP D,IL	Compares the value of the D register with that of the 16 lower order bits of the I register and sets the flags accordingly.
04E0	CMP D,M	Compares the value of the D register with that of the memory and sets the flags accordingly.
04E1	CMP E,F	Compares the value of the E register with that of the F register and sets the flags accordingly.
04E2	CMP E,GH	Compares the value of the E register with that of the 16 higher order bits of the G register and sets the flags accordingly.
04E3	CMP E,GL	Compares the value of the E register with that of the 16 lower order bits of the G register and sets the flags accordingly.
04E4	CMP E,HH	Compares the value of the E register with that of the 16 higher order bits of the H register and sets the flags accordingly.
04E5	CMP E,HL	Compares the value of the E register with that of the 16 lower order bits of the H register and sets the flags accordingly.
04E6	CMP E,IH	Compares the value of the E register with that of the 16 higher order bits of the I register and sets the flags accordingly.
04E7	CMP E,IL	Compares the value of the E register with that of the 16 lower order bits of the I register and sets the flags accordingly.
04E8	CMP E,M	Compares the value of the E register with that of the memory and sets the flags accordingly.
04E9	CMP F,GH	Compares the value of the F register with that of the 16 higher order bits of the G register and sets the flags accordingly.
04EA	CMP F,GL	Compares the value of the F register with that of the 16 lower order bits of the G register and sets the flags accordingly.
04EB	CMP F,HH	Compares the value of the F register with that of the 16 higher order bits of the H register and sets the flags accordingly.
04EC	CMP F,HL	Compares the value of the F register with that of the 16 lower order bits of the H register and sets the flags accordingly.
04ED	CMP F,IH	Compares the value of the F register with that of the 16 higher order bits of the I register and sets the flags accordingly.
04EE	CMP F,IL	Compares the value of the F register with that of the 16 lower order bits of the I register and sets the flags accordingly.
04EF	CMP F,M	Compares the value of the F register with that of the memory and sets the flags accordingly.
04F0	CMP GH,GL	Compares the value of the 16 higher order bits of the G register with that of the 16 lower order bits of the G register and sets the flags accordingly.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
04F1	CMP GH,HH	Compares the value of the 16 higher order bits of the G register with that of the 16 higher order bits of the H register and sets the flags accordingly.
04F2	CMP GH,HL	Compares the value of the 16 higher order bits of the G register with that of the 16 lower order bits of the H register and sets the flags accordingly.
04F3	CMP GH,IH	Compares the value of the 16 higher order bits of the G register with that of the 16 higher order bits of the I register and sets the flags accordingly.
04F4	CMP GH,IL	Compares the value of the 16 higher order bits of the G register with that of the 16 lower order bits of the I register and sets the flags accordingly.
04F5	CMP GH,M	Compares the value of the 16 higher order bits of the G register with that of the memory and sets the flags accordingly.
04F6	CMP GL,HH	Compares the value of the 16 lower order bits of the G register with that of the 16 higher order bits of the H register and sets the flags accordingly.
04F7	CMP GL,HL	Compares the value of the 16 lower order bits of the G register with that of the 16 lower order bits of the H register and sets the flags accordingly.
04F8	CMP GL,IH	Compares the value of the 16 lower order bits of the G register with that of the 16 higher order bits of the I register and sets the flags accordingly.
04F9	CMP GL,IL	Compares the value of the 16 lower order bits of the G register with that of the 16 lower order bits of the I register and sets the flags accordingly.
04FA	CMP GL,M	Compares the value of the 16 lower order bits of the G register with that of the memory and sets the flags accordingly.
04FB	CMP HH,HL	Compares the value of the 16 higher order bits of the H register with that of the 16 lower order bits of the H register and sets the flags accordingly.
04FC	CMP HH,IH	Compares the value of the 16 higher order bits of the H register with that of the 16 higher order bits of the I register and sets the flags accordingly.
04FD	CMP HH,IL	Compares the value of the 16 higher order bits of the H register with that of the 16 lower order bits of the I register and sets the flags accordingly.
04FE	CMP HH,M	Compares the value of the 16 higher order bits of the H register with that of the memory and sets the flags accordingly.
04FF	CMP HL,IH	Compares the value of the 16 lower order bits of the H register with that of the 16 higher order bits of the I register and sets the flags accordingly.
0500	CMP HL,IL	Compares the value of the 16 lower order bits of the H register with that of the 16 lower order bits of the I register and sets the flags accordingly.
0501	CMP HL,M	Compares the value of the 16 lower order bits of the H register with that of the memory and sets the flags accordingly.
0502	CMP IH,IL	Compares the value of the 16 higher order bits of the I register with that of the 16 lower order bits of the I register and sets the flags accordingly.
0503	CMP IH,M	Compares the value of the 16 higher order bits of the I register with that of the memory and sets the flags accordingly.
0504	CMP IL,M	Compares the value of the 16 lower order bits of the I register with that of the memory and sets the flags accordingly.
0505	CMI data	Compares the value of the data with that stored in the accumulator and sets the flags accordingly.
0506	CPP A,C	Compares the A/B register pair and the C/D register pair and sets the flags accordingly.
0502	CPP A,E	Compares the A/B register pair and the E/F register pair and sets the flags accordingly.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
0503	CPP A,G	Compares the A/B register pair and the G register and sets the flags accordingly.
0504	CPP A,H	Compares the A/B register pair and the H register and sets the flags accordingly.
0505	CPP A,I	Compares the A/B register pair and the I register and sets the flags accordingly.
0506	CPP A,SP	Compares the A/B register pair and the SP and sets the flags accordingly.
0503	CPP C,E	Compares the C/D register pair and the E/F register pair and sets the flags accordingly.
0504	CPP C,G	Compares the C/D register pair and the G register and sets the flags accordingly.
0505	CPP C,H	Compares the C/D register pair and the H register and sets the flags accordingly.
0506	CPP C,I	Compares the C/D register pair and the I register and sets the flags accordingly.
0504	CPP C,SP	Compares the C/D register pair and the SP and sets the flags accordingly.
0505	CPP E,G	Compares the E/F register pair and the G register and sets the flags accordingly.
0506	CPP E,H	Compares the E/F register pair and the H register and sets the flags accordingly.
0507	CPP E,I	Compares the E/F register pair and the I register and sets the flags accordingly.
0508	CPP E,SP	Compares the E/F register pair and the SP and sets the flags accordingly.
0509	CPP G,H	Compares the G register and the H register and sets the flags accordingly.
050A	CPP G,I	Compares the G register and the I register and sets the flags accordingly.
050B	CPP G,SP	Compares the G register and the SP and sets the flags accordingly.
050C	CPP H,I	Compares the H register and the I register and sets the flags accordingly.
050D	CPP H,SP	Compares the H register and the SP and sets the flags accordingly.
050E	CPP I,SP	Compares the I register and the SP and sets the flags accordingly.
050F	DAD C	Adds the contents of C/D register pair to those of I register and stores the result in I register. Carry Flag is modified.
0510	DAD E	Adds the contents of E/F register pair to those of I register and stores the result in I register. Carry Flag is modified.
0511	DAD G	Adds the contents of G register to those of I register and stores the result in I register. Carry Flag is modified.
0512	DAD H	Adds the contents of H register to those of I register and stores the result in I register. Carry Flag is modified.
0513	DAD SP	Adds the contents of SP to those of I register and stores the result in I register. Carry Flag is modified.
0514	SBD C	Subtracts the value stored at C/D Register Pair from the I register. Borrow Flag is modified.
0515	SBD E	Subtracts the value stored at E/F Register Pair from the I register. Borrow Flag is modified.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
0516	SBD G	Subtracts the value stored at G register from the I register. Borrow Flag is modified.
0517	SBD H	Subtracts the value stored at H register from the I register. Borrow Flag is modified.
0518	SBD SP	Subtracts the value stored at SP from the I register. Borrow Flag is modified.
0519	SBDI C	Subtracts the value stored at I register from the C/D Register Pair. Borrow Flag is modified.
051A	SBDI E	Subtracts the value stored at I register from the E/F Register Pair. Borrow Flag is modified.
051B	SBDI G	Subtracts the value stored at I register from the G register. Borrow Flag is modified.
051C	SBDI H	Subtracts the value stored at I register from the H register. Borrow Flag is modified.
051D	SBDI SP	Subtracts the value stored at I register from the SP. Borrow Flag is modified.
0700	CALL address	Calls a subroutine starting from the address mentioned. The contents of the registers are saved in the stack pointed by CP.
0701	CC address	Calls a subroutine starting from the address mentioned if Carry bit is set. The contents of the registers are saved in the stack pointed by CP.
0702	CNC address	Calls a subroutine starting from the address mentioned if Carry bit is reset. The contents of the registers are saved in the stack pointed by CP.
0703	CZ address	Calls a subroutine starting from the address mentioned if Zero bit is set. The contents of the registers are saved in the stack pointed by CP.
0704	CNZ address	Calls a subroutine starting from the address mentioned if Zero bit is reset. The contents of the registers are saved in the stack pointed by CP.
0705	CEP address	Calls a subroutine starting from the address mentioned if parity is even. The contents of the registers are saved in the stack pointed by CP.
0706	COP address	Calls a subroutine starting from the address mentioned if parity is odd. The contents of the registers are saved in the stack pointed by CP.
0707	CP address	Calls a subroutine starting from the address mentioned if it is positive. The contents of the registers are saved in the stack pointed by CP.
0708	CN address	Calls a subroutine starting from the address mentioned if it is negative. The contents of the registers are saved in the stack pointed by CP.
0709	CE address, data	Calls a subroutine starting from the address mentioned if the value stored in the accumulator is equal to the data provided. The contents of the registers are saved in the stack pointed by CP.
070A	CNE address, data	Calls a subroutine starting from the address mentioned if the value stored in the accumulator is not equal to the data provided. The contents of the registers are saved in the stack pointed by CP.
070B	CG address, data	Calls a subroutine starting from the address mentioned if the value stored in the accumulator is greater than the data provided. The contents of the registers are saved in the stack pointed by CP.
070C	CL address, data	Calls a subroutine starting from the address mentioned if the value stored in the accumulator is less than the data provided. The contents of the registers are saved in the stack pointed by CP.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
070D	RET	It returns the program back to the main program from the subroutine. The contents of the CP are POPed into the corresponding registers. CALL command must be followed by a RET statement.
070E	RC	It returns the program back to the main program from the subroutine if Carry bit is set. The contents of the CP are POPed into the corresponding registers. CALL command must be followed by a RET statement.
070F	RNC	It returns the program back to the main program from the subroutine if Carry bit is reset. The contents of the CP are POPed into the corresponding registers. CALL command must be followed by a RET statement.
0710	RZ	It returns the program back to the main program from the subroutine if Zero bit is set. The contents of the CP are POPed into the corresponding registers. CALL command must be followed by a RET statement.
0711	RNZ	It returns the program back to the main program from the subroutine if Zero bit is reset. The contents of the CP are POPed into the corresponding registers. CALL command must be followed by a RET statement.
0712	REP	It returns the program back to the main program from the subroutine if parity is even. The contents of the CP are POPed into the corresponding registers. CALL command must be followed by a RET statement.
0713	ROP	It returns the program back to the main program from the subroutine if parity is odd. The contents of the CP are POPed into the corresponding registers. CALL command must be followed by a RET statement.
0714	RP	It returns the program back to the main program from the subroutine if it is positive. The contents of the CP are POPed into the corresponding registers. CALL command must be followed by a RET statement.
0715	RN	It returns the program back to the main program from the subroutine if it is negative. The contents of the CP are POPed into the corresponding registers. CALL command must be followed by a RET statement.
0716	RE data	It returns the program back to the main program from the subroutine if the value stored in the accumulator is equal to the provided data. The contents of the CP are POPed into the corresponding registers. CALL command must be followed by a RET statement.
0717	RNE data	It returns the program back to the main program from the subroutine if the value stored in accumulator is not equal to the provided data. The contents of the CP are POPed into the corresponding registers. CALL command must be followed by a RET statement.
0718	RG data	It returns the program back to the main program from the subroutine if the value stored in accumulator is greater than the provided data. The contents of the CP are POPed into the corresponding registers. CALL command must be followed by a RET statement.
0719	RL data	It returns the program back to the main program from the subroutine if the value stored in accumulator is less than the provided data. The contents of the CP are POPed into the corresponding registers. CALL command must be followed by a RET statement.
071D	JMP address	It jumps the program unconditionally to the address given.
071E	JC address	It jumps the program if Carry Flag is set to the address given.
071F	JNC address	It jumps the program if Carry Flag is reset to the address given.
0720	JZ address	It jumps the program if Zero Flag is set to the address given.

- Memory implies the address indicated by I register pair.

Opcode	Mnemonics	Comments
0721	JNZ address	It jumps the program if Zero Flag is reset to the address given.
0722	JEP address	It jumps the program if parity is even to the address given.
0723	JOP address	It jumps the program if parity is odd to the address given.
0724	JP address	It jumps the program if it is positive to the address given.
0725	JN address	It jumps the program if it is negative to the address given.
0726	JE address, data	It jumps the program to the address given if the value of the accumulator is equal to the provided data.
0727	JNE address, data	It jumps the program to the address given if the value of the accumulator is not equal to the provided data.
0728	JG address, data	It jumps the program to the address given if the value of the accumulator is greater than the provided data.
0729	JL address, data	It jumps the program to the address given if the value of the accumulator is less than the provided data.
072A	IN address	Copies the 16-bit data from the I/O address to the accumulator.
072B	OUT address	Copies the contents of the accumulator to the I/O address.

- Memory implies the address indicated by I register pair.