ANC16: 16-bit architecture

Introduction

The ANC16 architecture was created for educational purposes. It is an easy architecture useful for studying.

Abbreviations

- ARC: architecture.
- INS: instruction.
- A: accumulator register.
- B: base register.
- SP: stack pointer.
- GPR: general purpose register.
- IO: input/output.
- DR: decremental register.
- SR: status register.
- PC: program counter.
- IRQ: interrupt request.
- DR-INR: interrupt request issued by DR.
- EIRQ: external IRQ.
- NMI: non-maskable interrupt.
- ISA: instruction set architecture.
- OS: operative system.
- MSB: most significant byte (8-bit).
- LSB: least significant byte.
- HEX: hexadecimal.
- BIN: binary.
- OCT: octal
- OPC: operation code
- INR: interrupt

Architecture

An ANC16 microprocessor is composed of different things. It has 16-bit and 8-bit length registers, a 64KB integrated random access memory, a 512B integrated ROM, the ALU and different buses.

Registers

A register is a tiny memory used to temporary store data such as addresses, numbers or characters to print on the screen.

There are different registers for different purposes, we call the registers used for general purposes GPR.

There are 4 GPRs and others registers with specific purposes. We list below the 4 GPRs

- The **A**ccumulator is a 16-bit GPR, divided in **A**ccumulator **H**igh (8-bit) and **A**ccumulator **L**ow (8-bit)
- The **B**ase is another 16-bit GPR, divided in **B**ase **H**igh (8-bit) and **B**ase **L**ow (8-bit)
- The **I**ndex register is a 16-bit GPR usually used to store addresses
- And finally the **J**ump To register that is the only 8-bit GPR usually used to store **relative addresses**

Summarizing, there are 4 GPRs: A, B, I and J. I and J registers are GPRs but are usually used to store addresses.

There are others registers with different purposes:

- The **P**rogram **C**ounter is a 16-bit register used to store the address of the current instruction
- The **I**nstruction **R**egister is a 16-bit read-only register used to store the current instruction
- The **S**tack **P**ointer is a 16-bit register used to refer to a certain address of an area of the memory used as stack.
- The **S**tatus **R**egister is an 8-bit register used to store additional information about the result of the current instruction:
    - **N**egative: 1 the result is negative, otherwise 0.
    - **O**verflow: 1 the result is out of the range -32.768, 32.767.
    - **I**nterrupts: 1 interrupts are enabled, 0 disabled (NMI, SYS and RESET interrupts are always enabled).
    - **D**R interrupts: 1 DR-INR is enabled, 0 disabled (when is disabled the register keep counting anyway).

    - **S**ystem privileges: 1 if system privileges are enable, otherwise 0. This flag can be modified only if is set to 1, otherwise is read-only.
    - NOT USED, always 1.
    - **Z**ero: 1 if the result is 0, otherwise 0.
    - **C**arry: 1 if the result is out of range 0, 65.535
- The **D**ecremental **R**egister is an 8-bit register that decrements itself at each cycle if it is 00 and the DR interrupt flag is 1, then a DR-INR is issued.
- The **I**nternal **M**emory **L**ower **I**ndex and **I**nternal **M**emory **H**igher **I**ndex are two 16-bit long registers that are used by the OS to specify the memory space available for an application running where IMLI is the starting address (included) and IMHI is the ending address (excluded). These two register are accessible only if the value in the **System Privileges** are enable.
- The **E**xternal **M**emory **L**ower **I**ndex and **E**xternal **M**emory **H**igher **I**ndex are two 16-bit long registers that have the same function as registers **IMLI** and **IMHI** but with external memory. These two register are accessible only if the value in the **System Privileges** are enable.
- The **A**ddress **R**egister is a 16-bit read-only register that contains the processed address argument of instructions that support internal/external memory read and write operations. For processed address we refer to an absolute address where the argument is fetched.

Buses

A bus is a connection between internal CPU's components. There are 3 buses with different widths.

- Address bus: 16-bit used to specify an address of a memory cell or an IO device, a cell stores 8-bit, a byte.
- Data bus: 16-bit used to transfer data among microprocessor’s components.
- Control bus: a 4-bit length bus that is used to specify the operation such as read or write. The first bit is used to specify internal or external R/W. The second bit is used to specify read (0) or write (1). The third is the enable, and the last one specifies the data width (0 = 8-bit, 1 = 16-bit)

Instructions

An instruction is a number that tells the ALU which operation should perform. An instruction is 16-bit long. There are ins. that takes no parameters and others that may take different parameters. Instructions are of different categories:

- Arithmetic operations, such as the sum, or the subtraction, increment, decrement

- Logical operations, such as AND, OR
- Shift operations, such as shift left or shift right
- Transfer / Load instructions, used to transfer data from registers or memory
- Stack instructions, used to perform stack operations.
- Flag instructions, that change or read the **SR** (Status Register).
- Comparison instructions, used to compare data.
- Jumps, that change the **PC** (Program Counter).
- Interrupt instructions, used to handle interrupts.
- IO instructions.

Memory

The 64KB integrated memory is used to store the current program in execution, data, or information regarding IO devices.

The memory is mapped as follows:

- From 0000 to 00FF there is the zero page.
- From 0100 to 1FFF there is the memory reserved to operative system routines.
- From 2000 to 2001 there is the OS ENTRY POINT vector, that store the entry point of the operative system
- From 2002 to 2003 there is the **DR-INR** vector, that store the 16-bit (2 bytes) address of the routine that handles the interrupt request issued by the **DR** (Decremental Register).
- From 2004 to 2005 there is the **EIRQ** vector, that store the address of the routine that handles the external interrupt request.
- From 2006 to 2007 there is the **NMI** vector, that store the routine address that handles a non-maskable interrupt.
- From 2008 to 2009 there is the SYSCALL vector, where is stored the address of the routine that handles system calls from applications.
- From 200A to 200B there is the **IAOOR-INR** vector, that store the 16-bit (word) address of the routine that handles the interrupt issued when the **AR** is Out Of Range (the range is defined by IMLI and IMHI registers).
- From 200C to 200D there is the **EAOOR-INR** vector, that store the 16-bit (word) address of the routine that handles the interrupt issued when the **AR** is Out Of Range (the range is defined by EMLI and EMHI registers).

- From 200E to 200F there is the **SPOOR-INR** vector, that store the address of the routine that handles the interrupt issued when stack overflow occurs
- From 2010 to 2011 there is the vector used to store the **PC** when an irq is issued.
- 2012 there is the vector used to store the **SR** when an irq is issued.
- From 2013 to 2014 there is the vector used to store the **A** register when an irq is issued.
- From 2015 to 2016 there is the vector used to store the **B** register when an irq is issued.
- From 2017 to 34FF (arbitrary) there is the memory reserved to operative system data.
- From 3500 (arbitrary) to FDFF free memory.
- From FE00 to FFFF ROM with firmware.

The format of an address of a cell in memory is PPAA where PP is a byte that refers to the page, AA is the address in the page.

ROM

The 512B Read Only Memory contains the firmware that loads the operative system into memory from an external ROM, the starting loading address is 0000 and the ending address is 200B (included) (IO Address).

Addressing mode

There are different ways of referring to a position in memory.

- Absolute: when the argument of the instruction is the address of a cell in memory
- Absolute indexed: from the absolute address, is added **I** as signed integer
- Relative: when the argument of the instruction is the address of a cell calculated by adding the 8-bit signed value to the **PC**.
- Relative with J: as the Relative, with the difference that there is no argument, the value added to the PC is the value stored in **J**.
- Indirect: when the argument of the instruction is an address stored in a cell in memory referred by an absolute address.
- Indirect indexed: from the indirect address, is added **I** as signed integer to the final address.
- Implied: when an instruction takes no argument.
- Immediate: when the argument is the operand.
- Zero page: when the argument is a byte that refers to the first page of the integrated memory.
- Zero page indexed: from the zero page, is added **I** as unsigned integer.

- Accumulator: when the operand is the **A** register or **AH** or **AL**.
- Base: when the operand is the **B** register, **BH** or **BL**.
- Index register: when the operand is stored in **I**.

Interrupts

An interrupt is an internal or external signal that interrupt the execution of the CPU and start a routine called interrupt handler.

Hardware interrupts:

- EIRQ: maskable external interrupt request. Address to routine stored in 2004 – 2005 (hex). The address is stored in **B** and data are stored in **AH**.
- NMI: non-maskable interrupt. Address to routine stored in 2006 – 2007 (in hex).
  When the CPU does not recognize an opcode a NMI is issued and in **AL** is stored 1.
  When an application tries to execute an instruction that require **System Privileges** a NMI is issued and in **AL** is stored 2.
- *RESET: restart the CPU, also a software interrupt.
- DR-INR: this interrupt is issued when the register **DR** is 0000. Address to interrupt handler is stored in 2002 – 2003 (hex)
- **IAOOR-INR: (**I**nternal **A**ddress **O**ut **O**f **R**ange **In**te**r**rupt) this interrupt is issued when the **AR** is out of the range from the value stored in **IMLI** (included) to the value stored in **IMHI** (included). Address to interrupt handler is stored in 200A – 200B.
- **EAOOR-INR: (**E**xternal **A**ddress **O**ut **O**f **R**ange **In**te**r**rupt) this interrupt is issued when the **AR** is out of the range from the value stored in **EMLI** (included) to the value stored in **EMHI** (included). Address to interrupt handler is stored in 200C – 200D.
- **SPOOR-INR: this interrupt is very similar to **AOOR-INR**. The **SP** **O**ut **O**f **R**ange inr. is issued when the **SP** is lower than the value stored in **IMLI** or is higher than the value stored in **IMHI**.

Software interrupt

- SYS: is a software interrupt used for system calls. Address to routine is stored in 2008 – 2009

Interrupts marked with * do not store **PC** and **SR**.

Interrupts marked with ** are issued only if **System Privileges** is not set.

When an interrupt (except for RESET) is issued the SR changes into n o I d S 1 z c (uppercase = 1)

Instruction Set Architecture

The instruction set architecture, or ISA, is the set of instructions recognized by the microprocessor. We show the summary table of the instructions.

Instruction marked with * can be executed only if **System Privileges** is set.

#	Mnemonic	Description	Type
1	ADA	Add A and operand and store the result in A	Arithmetic
2	ADB	Add B and operand and store the result in B	Arithmetic
3	ANA	A AND operand	Logical
4	ANB	B AND operand	Logical
5	ARET	Zero flag set if A contains the RET instruction code	Flag
6	CLC	Set Carry flag in the SR to 0	Flag
7	*CLD	Set DR Interrupt flag in the SR to 0	Flag
8	*CLI	Set Interrupt flag in the SR to 0	Flag
9	CLO	Set Overflow flag in the SR to 0	Flag
10	*CLS	Set System privileges in the SR to 0	Flag
11	CMAH	Compare AH to operand	Compare
12	CMBH	Compare BH to operand	Compare
13	CMPA	Compare A to operand	Compare
14	CMPB	Compare B to operand	Compare
15	CMPI	Compare I to operand	Compare
16	CPUID	Store the CPU ID in registers	Other
17	DEA	Decrement A	Arithmetic
18	DEB	Decrement B	Arithmetic
19	DEI	Decrement I	Arithmetic
20	DEJ	Decrement J	Arithmetic
21	INA	Increment A	Arithmetic
22	INB	Increment B	Arithmetic
23	INI	Increment I	Arithmetic
24	INJ	Increment J	Arithmetic
25	JCC	Jump if Carry is set to 0	Jump
26	JCS	Jump if Carry is set to 1	Jump
27	JEQ	Jump if equal (Zero is set)	Jump
28	JMP	Change the PC	Jump
29	JNC	Jump if Negative is set to 0	Jump
30	JNE	Jump if not equal (Zero is clear)	Jump
31	JNS	Jump if Negative is set to 1	Jump
32	JOC	Jump if Overflow is set to 0	Jump
33	JOS	Jump if Overflow is set to 1	Jump
34	*KILL	Stop the CPU	Software IRQ
35	LDA	Load A	Load
36	LDAH	Load AH	Load
37	LDAL	Load AL	Load
38	LDB	Load B	Load
39	LDBH	Load BH	Load
40	LDBL	Load BL	Load
41	*LDDR	Load DR	Load

42	LDI	Load I	Load
43	LDJ	Load J	Load
44	LDSP	Load Stack Pointer	Load
45	LDSR	Load SR register	Load
46	*LEMH	Load EMHI register	Load
47	*LEML	Load EMLI register	Load
48	*LIMH	Load IMHI register	Load
49	*LIML	Load IMLI register	Load
50	MSB	Check if the most significant bit in A or B is 0 or 1 (the result is stored in zero flag)	Other
51	NOP	No operation	Other
52	ORA	A OR operand	Logical
53	ORB	B OR operand	Logical
54	POP	Pop from the stack and update the SP	Stack
55	PSH	Push onto the stack and update the SP, used to CALL routines	Stack
56	READ	Read from IO Devices	IO
57	*REST	Restart interrupt	Software IRQ
58	RET	Return from a routine	Stack
59	*SED	Set DR Interrupt flag in the SR to 1	Flag
60	*SEI	Set Interrupt flag in the SR to 1	Flag
61	SEMH	Store EMHI register	Strore
62	SEML	Store EMLI register	Store
63	*SES	Set System Privileges in the SR to 1	Flag
64	SHL	Shift left A or B, update the Carry	Shift
65	SHR	Shift right A or B, update the Carry	Shift
66	SIMH	Store IMHI register	Store
67	SIML	Store IMLI register	Store
68	STA	Store A in memory	Store
69	STAH	Save AH in memory	Store
70	STB	Store B in memory	Store
71	STBH	Save BH in memory	Store
72	STI	Store I in memory	Store
73	STJ	Store J in memory	Store
74	STPC	Store PC	Store
75	STSR	Store SR	Store
76	SUA	Subtract A and operand and store the result in A	Arithmetic
77	SUB	Subtract B and operand and store the result in B	Arithmetic
78	SYS	Call to System interrupt	Software IRQ
79	TAB	Transfer A to B	Transfer
80	TABH	Transfer AH to BH	Transfer
81	TABL	Transfer AL to BL	Transfer
82	*TADR	Transfer A to DR	Transfer
83	*TAEMH	Transfer A to EMHI	Transfer
84	*TAEML	Transfer A to EMLI	Transfer
85	TAHJ	Transfer AH to J	Transfer
86	TAI	Transfer A to I	Transfer
87	*TAIMH	Transfer A to IMHI	Transfer
88	*TAIML	Transfer A to IMLI	Transfer

89	TBA	Transfer B to A	Transfer
90	TBAH	Transfer BH to AH	Transfer
91	TBAL	Transfer BL to AL	Transfer
92	TBHJ	Transfer BH to J	Transfer
93	TBI	Transfer B to I	Transfer
94	TISP	Transfer I to Stack Pointer	Transfer
95	TSPB	Transfer Stack Pointer to B	Transfer
96	WRTE	Write into IO Devices, Data are stored in AH	IO
97	WRTI	Write into IO Devices, Address is stored in I	IO
98	XORA	A XOR operand	Logical
98	XORB	B XOR operand	Logical

Instruction marked with * can be executed only if **System Privileges** is set.

Assembly standard

This tells how to write in ANC16 Assembly:

- HEX representation: **0xHHHH**… where H is a hex digit (0 – F).
- BIN representation: **0bBBBB**… where B is 0 or 1.
- OCT representation: **0oOOOO**… where O is an oct digit (0 – 7).
- Decimal representation: **DDDD**… where D is a decimal digit (0 – 9).
- Absolute addressing example: READ **0xFF00**.
- Absolute indexed: STA **0xFF00, I.**
- Relative: JMP ***0b10**.
- Relative with J: JNS ***J**.
- Indirect: JMP **[0xFF00]**.
- Indirect indexed: JMP **[0xFF00], I**.
- Implied: SYS.
- Immediate: LDA **#0xFF00**.
- Zero page: ADA **%0xFF**.
- Zero page indexed: SUB **%0xFF, I**.
- Accumulator: SHL **A**.
- Base: SHR **B**.

Directives and predefined routines:

The list shown below may differ from assembler to assembler:

- **ORG**: set the starting address of a label.

- **USE STDCALL**: this allows you to use CALL predefined routine.
- **USE AS**: this allows you to define constants: USE six AS 6.
- **IMPORT:** this allows you to import a library.
- **WORD:** this allows you to specify if the next number is 2 bytes long.
- **BYTE:** the same as WORD, but the next number is just 1 byte long.
- **CALL:** this is a macro that allows you to call a routine simply by using the label name.
- **SYSCALL:** this is a macro used to make a system call referring to the syscall name.

System calls

A system call is a software interrupt managed by the operating system and is used by programs to read and modify resources that only the operating system can access such as video memory. The system call inr. is issued using the **SYS** instruction. Arguments are stored in the registers. The system call code is saved in the register **AL**.

This is the list of standard system calls:

exit

The exit system call is used to kill the execution of the program.

- Code: 0x00
- Arguments
  - **BL** register: exit status code (0x00 is no error)

fopen

Is used to open files or streams.

- Code: 0x01
- Arguments
  - **B** register: the address of the string that represents the file path.
  - **I** register: the address to a cell in memory (8-bit) that will contains the file descriptor id
- Return
  - **AH** register: 0x01 in case of error

fclose

Is used to close and save files.

- Code: 0x02

- Arguments
  - AH register: the file descriptor id
- Return
  - AH register: 0x01 in case of error

fread

Is used to read from streams.

- Code: 0x03
- Arguments
  - AH register: the file descriptor id
  - BL register: the buffer size
  - I register: the pointer to the buffer

fwrite

Is used write in streams.

- Code: 0x04
- Arguments
  - AH register: the file descriptor id
  - BH: the write mode (0 = append, 1 = truncate)
  - BL: the buffer size
  - I: the pointer to the buffer that contains the content.

print

Is used to print strings in the standard output stream.

- Code: 0x05
- Arguments
  - BL register: the length of the string
  - I register: the pointer to the buffer

getl

Is used to get lines from the standard input stream.

- Code: 0x06
- Arguments
  - BL register: the length of the destination buffer
  - I register: the pointer to the destination buffer

sleep

The sleep system call is used to pause the execution of the program.

- Code: 0x07
- Arguments
  - B register: number of cycle to sleep

listenKey

Is used to handle keyboard events.

- Code: 0x08
- Arguments
  - I register: the address to the event handler procedure
- Return
  - AH: the key code (used by the event handler procedure)

requestPrivileges

Is used to request System Privileges.

- Code: 0x09
- Return
  - AH: 0 = not allowed, 1 = allowed

malloc

Is used to allocate memory dynamically.

- Code: 0x0A
- Arguments
  - BL: the memory size.
- Return
  - I: the address of the allocated area or 0x0000 in case of error.

dealloc

Is used to free memory dynamically.

- Code: 0x0B
- Arguments
  - I: the address of the area.
- Return
  - AH: 0x01 in case of error.

On Reset

When the CPU is restarted or turned on, the PC is set to FE00, where the firmware resides. The SR is set to: n o I D S 1 z c (uppercase = 1)

Variants

There are different variants:

- **1J** (code in hex 01): "Jump To" register is 16-bit long
- **1NZ** (code in hex 80): Not zero page addressing
- **1JNZ** (code in hex 81): As 1J with 1NZ
- **1JR** (code in hex 02): As 1J with but the CPU has 2 more GPRs, R1 (16-bit long) and R2 (16-bit long)
- **1JRNZ** (code in hex 82): As 1JR with 1NZ

The standard variant has 00 as code. The code is used by the CPUID instruction.

Instructions in detail

All 392 OPCs.

Instruction marked with * can be executed only if System Privileges is set.

ADA – ADd Accumulator

A = A + operand

This operation may change the flags: N, O, Z, C.

OPC in hex	Addressing mode	Argument length in bytes	Description
0014	Base register	0	A = A + B
4004	Immediate	1	AL = AL + argument (8-bit or byte)

4005	Zero page	1	A = A + word (16-bit) stored in the zero page
4006	Zero page indexed	1	A = A + word
8004	Absolute	2	A = A + word
8005	Absolute indexed	2	A = A + word
8006	Indirect	2	A = A + word
8007	Indirect indexed	2	A = A + word
8033	Immediate	2	A = A + argument (word)

ADB – ADd Base register

B = B + operand

This operation may change the flags: N, O, Z, C.

OPC in hex	Addressing mode	Argument length in bytes	Description
0015	Accumulator register	0	B = B + A
4007	Immediate	1	BL = BL + argument (byte)
4008	Zero page	1	B = B + word (16-bit) stored in the zero page
4009	Zero page indexed	1	B = B + word
8008	Absolute	2	B = B + word
8009	Absolute indexed	2	B = B + word
800A	Indirect	2	B = B + word
800B	Indirect indexed	2	B = B + word
8034	Immediate	2	B = B + argument (word)

ANA – ANd with Accumulator

A = A bit-wise and with operator

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
0016	Base register	0	A = A AND B
0018	Index register	0	A = A AND I
400A	Immediate	1	AL = AL AND argument (8-bit or byte)
400B	Zero page	1	A = A AND word (16-bit) stored in the zero page
400C	Zero page indexed	1	A = A AND word

800C	Absolute	2	A = A AND word
800D	Absolute indexed	2	A = A AND word
800E	Indirect	2	A = A AND word
800F	Indirect indexed	2	A = A AND word
8035	Immediate	2	A = A AND argument (word)

ANB – ANd with Base register

B = B bit-wise and with operator

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
0017	Accumulator register	0	B = B AND A
0019	Index register	0	B = B AND I
400D	Immediate	1	BL = BL AND argument (8-bit or byte)
400E	Zero page	1	B = B AND word (16-bit) stored in the zero page
400F	Zero page indexed	1	B = B AND word
8010	Absolute	2	B = B AND word
8011	Absolute indexed	2	B = B AND word
8012	Indirect	2	B = B AND word
8013	Indirect indexed	2	B = B AND word
8036	Immediate	2	B = B AND argument (word)

ARET – Accumulator contains RETurn

A == RET (in hex 80FF)

This operation may change the flags: Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
0002	Implied	0	A == RET

CLC – CLear Carry flag

Carry = 0

OPC in hex	Addressing mode	Argument length in bytes	Description
0010	Implied	0	Carry = 0

*CLD – CLear Dr interrupts flag

DR inr. = 0

OPC in hex	Addressing mode	Argument length in bytes	Description
0012	Implied	0	DR inr. = 0

*CLI – CLear Interrupt flag

Interrupt = 0

OPC in hex	Addressing mode	Argument length in bytes	Description
0013	Implied	0	Interrupt = 0

CLO – CLear Overflow flag

Overflow = 0

OPC in hex	Addressing mode	Argument length in bytes	Description
0011	Implied	0	Overflow = 0

*CLS – CLear System privileges

System Privileges = 0

OPC in hex	Addressing mode	Argument length in bytes	Description
0061	Implied	0	System Privileges = 0

CMAH – CoMpare AH to operand

AH == operand

Result (Register – operand)	Carry Flag	Zero Flag	Negative Flag
Register > operand	1	0	Sign bit of result
Register == operand	1	1	0
Register < operand	0	0	Sign bit of result

OPC in hex	Addressing mode	Argument length in bytes	Description

001A	BH register	0	AH == BH
403F	Immediate	1	AH == argument (byte)
4040	Zero page	1	AH == byte
4041	Zero page indexed	1	AH == byte
8014	Absolute	2	AH == byte
8015	Absolute indexed	2	AH == byte
8016	Indirect	2	AH == byte
8017	Indirect indexed	2	AH == byte

CMBH – CoMpare BH to operand

BH == operand

Result (Register – operand)	Carry Flag	Zero Flag	Negative Flag
Register > operand	1	0	Sign bit of result
Register == operand	1	1	0
Register < operand	0	0	Sign bit of result

OPC in hex	Addressing mode	Argument length in bytes	Description
001B	AH register	0	BH == AH
4042	Immediate	1	BH == argument (byte)
4043	Zero page	1	BH == byte
4044	Zero page indexed	1	BH == byte
8018	Absolute	2	BH == byte
8019	Absolute indexed	2	BH == byte
801A	Indirect	2	BH == byte
801B	Indirect indexed	2	BH == byte

CMPA – CoMpare Accumulator to operand

A == operand

Result (Register – operand)	Carry Flag	Zero Flag	Negative Flag
Register > operand	1	0	Sign bit of result
Register == operand	1	1	0
Register < operand	0	0	Sign bit of result

OPC in hex	Addressing mode	Argument length in bytes	Description
001C	Base register	0	A == B
001E	Index register	0	A == I
4045	Zero page	1	A == word (16-bit)
4046	Zero page indexed	1	A == word
801C	Absolute	2	A == word
801D	Absolute indexed	2	A == word
801E	Indirect	2	A == word
801F	Indirect indexed	2	A == word
8024	Immediate	2	A == argument (word)

CMPB – CoMpare Base register to operand

B == operand

Result (Register – operand)	Carry Flag	Zero Flag	Negative Flag
Register > operand	1	0	Sign bit of result
Register == operand	1	1	0
Register < operand	0	0	Sign bit of result

OPC in hex	Addressing mode	Argument length in bytes	Description
001D	Accumulator register	0	B == A
001F	Index register	0	B == I
4047	Zero page	1	B == word (16-bit)
4048	Zero page indexed	1	B == word
8020	Absolute	2	B == word
8021	Absolute indexed	2	B == word
8022	Indirect	2	B == word
8023	Indirect indexed	2	B == word
8025	Immediate	2	B == argument (word)

CMPI – CoMpare Index register to operand

I == operand

Result (Register – operand)	Carry Flag	Zero Flag	Negative Flag
Register > operand	1	0	Sign bit of result
Register == operand	1	1	0
Register < operand	0	0	Sign bit of result

OPC in hex	Addressing mode	Argument length in bytes	Description
0020	Accumulator register	0	I == A
4049	Zero page	1	I == word (16-bit)
8026	Immediate	2	I == word
8027	Absolute	2	I == word
8028	Indirect	2	I == word

CPUID – load CPU IDentifier

AL = ANC16 version (current: 1)

AH = Variant

OPC in hex	Addressing mode	Argument length in bytes	Description
0003	Implied	0	AL = ANC16 version (1)

DEA – DEcrement Accumulator

A = A – 1

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
0008	Implied	0	A = A – 1

DEB – DEcrement Base register

B = B – 1

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
0009	Implied	0	B = B – 1

DEI – DEcrement Index register

I = I – 1

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
000A	Implied	0	I = I – 1

DEJ – DEcrement "Jump to" register

J = J – 1

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
000B	Implied	0	J = J – 1

INA – INcrement Accumulator

A = A + 1

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
0004	Implied	0	A = A + 1

INB – INcrement Base register

B = B + 1

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
0005	Implied	0	B = B + 1

INI – INcrement Index register

I = I + 1

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
0006	Implied	0	I = I + 1

INJ – INcrement "Jump to" register

J = J + 1

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
0007	Implied	0	J = J + 1

JCC – Jump if Carry is Clear

Change PC if Carry flag (in Status Register) is 0

OPC in hex	Addressing mode	Argument length in bytes	Description
0021	Relative with J	0	C == 0 ? then PC = PC + J
4010	Relative	1	C == 0 ? then PC = PC + argument (byte)
8029	Absolute	2	C == 0 ? then PC = argument (word big endian)

JCS – Jump if Carry is Set

Change PC if Carry is 1

OPC in hex	Addressing mode	Argument length in bytes	Description
0025	Relative with J	0	C == 1 ? then PC = PC + J
4014	Relative	1	C == 1 ? then PC = PC + argument (byte)
802D	Absolute	2	C == 1 ? then PC = argument (word big endian)

JEQ – Jump if Equal (Zero is Set)

Change PC if Carry is 0

OPC in hex	Addressing mode	Argument length in bytes	Description
0027	Relative with J	0	Z == 1 ? then PC = PC + J
4016	Relative	1	Z == 1 ? then PC = PC + argument (byte)
802F	Absolute	2	Z == 1 ? then PC = argument (word big endian)

JMP – JuMP to

Change the PC

OPC in hex	Addressing mode	Argument length	Description

		in bytes	
0056	Accumulator register	0	PC = A
0057	Base register	0	PC = B
0058	Index register	0	PC = I
0063	Relative with J	0	PC = PC + J
405C	Relative	1	PC = PC + argument (byte)
8031	Absolute	2	PC = argument (word, big endian)
8032	Indirect	2	PC = word

JNC – Jump if Negative is Clear

Change PC if Negative flag is 0

OPC in hex	Addressing mode	Argument length in bytes	Description
0024	Relative with J	0	N == 0 ? then PC = PC + J
4013	Relative	1	N == 0 ? then PC = PC + argument (byte)
802C	Absolute	2	N == 0 ? then PC = argument (word big endian)

JNE – Jump if Not Equal (Zero is Clear)

Change PC if Zero is 0

OPC in hex	Addressing mode	Argument length in bytes	Description
0023	Relative with J	0	Z == 0 ? then PC = PC + J
4012	Relative	1	Z == 0 ? then PC = PC + argument (byte)
802B	Absolute	2	Z == 0 ? then PC = argument (word big endian)

JNS – Jump if Negative is Set

Change PC if Negative flag is 1

OPC in hex	Addressing mode	Argument length in bytes	Description
0028	Relative with J	0	N == 1 ? then PC = PC + J
4017	Relative	1	N == 1 ? then PC = PC + argument (byte)
8030	Absolute	2	N == 1 ? then PC = argument (word big endian)

JOC – Jump if Overflow is Clear

Change PC if Overflow flag is 0

OPC in hex	Addressing mode	Argument length in bytes	Description
0022	Relative with J	0	O == 0 ? then PC = PC + J
4011	Relative	1	O == 0 ? then PC = PC + argument (byte)
802A	Absolute	2	O == 0 ? then PC = argument (word big endian)

JOS – Jump if Overflow is Set

Change PC if Overflow is 1

OPC in hex	Addressing mode	Argument length in bytes	Description
0026	Relative with J	0	O == 1 ? then PC = PC + J
4015	Relative	1	O == 1 ? then PC = PC + argument (byte)
802E	Absolute	2	O == 1 ? then PC = argument (word big endian)

**KILL*

Stop the execution

OPC in hex	Addressing mode	Argument length in bytes	Description
0001	Implied	0	Stop the execution of the CPU
3FFD	Implied	0	Stop the execution of the CPU
8086	Absolute	2	Stop the execution of the CPU

LDA – LoaD Accumulator

A = memory

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
8037	Absolute	2	A = word
8038	Absolute indexed	2	A = word
8039	Indirect	2	A = word
803A	Indirect indexed	2	A = word
8053	Immediate	2	A = argument (word, big endian)

LDAH – LoaD Accumulator High

AH = memory

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
4018	Immediate	1	AH = argument (byte)
803B	Absolute	2	AH = byte
803C	Absolute indexed	2	AH = byte
803D	Indirect	2	AH = byte
803E	Indirect indexed	2	AH = byte

LDAL – LoaD Accumulator Low

AL = memory

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
4019	Immediate	1	AL = argument (byte)
803F	Absolute	2	AL = byte
8040	Absolute indexed	2	AL = byte
8041	Indirect	2	AL = byte
8042	Indirect indexed	2	AL = byte

LDB – LoaD Base register

B = memory

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
8043	Absolute	2	B = word
8044	Absolute indexed	2	B = word
8045	Indirect	2	B = word
8046	Indirect indexed	2	B = word
8054	Immediate	2	B = argument (word, big endian)

LDBH – LoaD Base register High

BH = memory

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
401A	Immediate	1	BH = argument (byte)

8047	Absolute	2	BH = byte
8048	Absolute indexed	2	BH = byte
8049	Indirect	2	BH = byte
804A	Indirect indexed	2	BH = byte

LDBL – LoaD Base register Low

BL = memory

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
401B	Immediate	1	BL = argument (byte)
804B	Absolute	2	BL = byte
804C	Absolute indexed	2	BL = byte
804D	Indirect	2	BL = byte
804E	Indirect indexed	2	BL = byte

**LDDR – LoaD Decremental Register*

DR = memory

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
808A	Absolute	2	DR = word
808B	Indirect	2	DR = word
808C	Immediate	2	DR = argument

LDI – LoaD Index register

I = memory

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
804F	Absolute	2	I = word
8050	Indirect	2	I = word
8055	Immediate	2	I = argument

LDJ – LoaD “Jump to” register

J = memory

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
401C	Immediate	1	J = argument (byte)
404A	Zero page	1	J = byte
8051	Absolute	2	J = byte
8052	Indirect	2	J = byte

LDSP – LoaD Stack pointer

SP = memory

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
8056	Absolute	2	SP = word
8057	Indirect	2	SP = word
8058	Immediate	2	SP = argument

LDSR – LoaD SR register

SR = memory.

This operation may change the flags: N, Z

OPC in hex	Addressing mode	Argument length in bytes	Description
4053	Immediate	1	SR = byte
80A9	Absolute	2	SR = byte
80B0	Indirect	2	SR = word

**LEMH – Load EMHI register*

EMHI = memory only if System Privileges is set.

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
4054	Zero page	1	EMHI = word
4055	Zero page indexed	1	EMHI = word
80B1	Absolute	2	EMHI = word
80B2	Indirect	2	EMHI = word
80B3	Immediate	2	EMHI = argument

*LEML – Load EMLI register

EMLI = memory only if System Privileges is set.

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
4056	Zero page	1	EMLI = word
4057	Zero page indexed	1	EMLI = word
80B4	Absolute	2	EMLI = word
80B5	Indirect	2	EMLI = word
80B6	Immediate	2	EMLI = argument

*LIMH – Load IMHI register

IMHI = memory only if System Privileges is set.

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
404B	Zero page	1	IMHI = word
404C	Zero page indexed	1	IMHI = word
808D	Absolute	2	IMHI = word
808E	Indirect	2	IMHI = word
808F	Immediate	2	IMHI = argument

*LIML – Load IMLI register

IMLI = memory only if System Privileges is set.

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
404D	Zero page	1	IMLI = word
404E	Zero page indexed	1	IMLI = word
8090	Absolute	2	IMLI = word
8091	Indirect	2	IMLI = word
8092	Immediate	2	IMLI = argument

MSB – Most Significant Bit

Check if the most significant bit is 1 or 0, store the result in Zero flag

This operation may change the flags: Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
000C	Accumulator register	0	Z = most significant bit of A
000D	Base register	0	Z = most significant bit of B

NOP – No OPeration

No operation

OPC in hex	Addressing mode	Argument length in bytes	Description
0000	Implied	0	No operation
0029	Implied	0	No operation
002A	Implied	0	No operation
002B	Implied	0	No operation
002C	Implied	0	No operation
002F	Implied	0	No operation
0030	Implied	0	No operation
0033	Implied	0	No operation
0034	Implied	0	No operation
0035	Implied	0	No operation
0069	Implied	0	No operation
3FFC	Implied	0	No operation
4000	Immediate	1	No operation
4023	Immediate	1	No operation
4038	Zero page indexed	1	No operation
405D	Immediate	1	No operation
7FFF	Immediate	1	No operation
8000	Absolute	2	No operation
8076	Absolute indexed	2	No operation
8078	Indirect indexed	2	No operation
80BF	Absolute	2	No operation
80FE	Absolute	2	No operation

ORA – OR with Accumulator

A = A bit-wise or with operand

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
0036	Base register	0	A = A OR B
0038	Index register	0	A = A OR I
401D	Immediate	1	AL = AL OR argument (8-bit or byte)
401E	Zero page	1	A = A OR word (16-bit) stored in the zero page
401F	Zero page indexed	1	A = A OR word
8059	Absolute	2	A = A OR word
805A	Absolute indexed	2	A = A OR word
805B	Indirect	2	A = A OR word
805C	Indirect indexed	2	A = A OR word
8061	Immediate	2	A = A OR argument (word)

ORB – OR with Base register

B = B bit-wise or with operand

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
0037	Accumulator register	0	B = B OR A
0039	Index register	0	B = B OR I
4020	Immediate	1	BL = BL OR argument (8-bit or byte)
4021	Zero page	1	B = B OR word (16-bit) stored in the zero page
4022	Zero page indexed	1	B = B OR word
805D	Absolute	2	B = B OR word
805E	Absolute indexed	2	B = B OR word
805F	Indirect	2	B = B OR word
8060	Indirect indexed	2	B = B OR word
8062	Immediate	2	B = B OR argument (word)

POP

Pop from the stack.

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description

0065	Accumulator register	0	A = [SP]; SP = SP − 2
0066	Base register	0	B = [SP]; SP = SP − 2
0067	AH register	0	AH = [SP]; SP = SP − 1
0068	AL register	0	AL = [SP]; SP = SP − 1

PSH – PuSH

Push onto the stack and increment the SP

OPC in hex	Addressing mode	Argument length in bytes	Description
003A	Accumulator register	0	Push A; SP = SP + 2
003B	Base register	0	Push B; SP = SP + 2
003C	AH register	0	Push AH; SP = SP + 1
003D	AL register	0	Push AL; SP = SP + 1
0064	Implied	0	Push SR; Push PC + 4; SP = SP + 3
4001	Immediate	1	Push argument (byte); SP = SP + 1
4002	Zero page indexed	1	Push word; SP = SP + 2
4003	Zero page	1	Push word; SP = SP + 2
8001	Absolute	2	Push word; SP = SP + 2
8002	Indirect	2	Push word; SP = SP + 2
8003	Indirect indexed	2	Push word; SP = SP + 2

READ

Read from IO devices

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
003E	Accumulator register	0	Read @ address stored in A, store data in BH
003F	Base register	0	Read @ address stored in B, store data in AH
4024	Zero page	1	Read @ address x00argument, store data in AH
4025	Zero page indexed	1	Read @ address x00argument + I, store data in AH
8063	Absolute	2	Read @ address argument, store data in AH
8064	Absolute indexed	2	Read @ address argument + I, store data

			in AH

**REST – RESeT (or RESTart)*

Issue a restart interrupt

OPC in hex	Addressing mode	Argument length in bytes	Description
3FFE	Implied	0	Issue a restart interrupt

RET – RETurn from routine

PC = fetched from the address in the SP (word)

SR = fetched from the address + 2 in the SP (byte)

OPC in hex	Addressing mode	Argument length in bytes	Description
80FF	Implied	0	PC = First 2 bytes (big endian) in the stack; SR = third byte; SP = SP − 3

**SED – Set DR interrupt flag*

DR inr. = 1

OPC in hex	Addressing mode	Argument length in bytes	Description
000F	Implied	0	DR = 1

**SEI – Set Interrupt flag*

I = 1

OPC in hex	Addressing mode	Argument length in bytes	Description
000E	Implied	0	I = 1

SEMH – Store EMHI

Memory = EMHI

OPC in hex	Addressing mode	Argument length in bytes	Description
4058	Zero page	1	Memory = EMHI
4059	Zero page indexed	1	Memory = EMHI
80B7	Absolute	2	Memory = EMHI (big endian)
80B8	Absolute indexed	2	Memory = EMHI (as always big endian)
80B9	Indirect	2	Memory = EMHI
80BA	Indirect indexed	2	Memory = EMHI

SEML – Store EMLI

Memory = EMLI

OPC in hex	Addressing mode	Argument length in bytes	Description
405A	Zero page	1	Memory = EMLI
405B	Zero page indexed	1	Memory = EMLI
80BB	Absolute	2	Memory = EMLI (big endian)
80BC	Absolute indexed	2	Memory = EMLI (as always big endian)
80BD	Indirect	2	Memory = EMLI (as always big endian...)
80BE	Indirect indexed	2	Memory = EMLI

**SES – Set System privileges*

System Privileges = 1

OPC in hex	Addressing mode	Argument length in bytes	Description
0062	Implied	0	System Privileges = 1

SHL – SHift Left

Operand << 1

This operation may change the flags: Z, C.

OPC in hex	Addressing mode	Argument length in bytes	Description
0040	Accumulator register	0	A << 1; update Carry
0041	Base register	0	B << 1; update Carry
0042	Index register	0	I << 1; update Carry

SHR – SHift Right

Operand >> 1

This operation may change the flags: Z, C.

OPC in hex	Addressing mode	Argument length in bytes	Description
0043	Accumulator register	0	A >> 1; update Carry
0044	Base register	0	B >> 1; update Carry
0045	Index register	0	I >> 1; update Carry

SIMH – Store IMHI

Memory = IMHI

OPC in hex	Addressing mode	Argument length in bytes	Description
404F	Zero page	1	Memory = IMHI
4050	Zero page indexed	1	Memory = IMHI
8093	Absolute	2	Memory = IMHI (big endian)
8094	Absolute indexed	2	Memory = IMHI (as always big endian)
8095	Indirect	2	Memory = IMHI
8096	Indirect indexed	2	Memory = IMHI

SIML – Store IMLI

Memory = IMLI

OPC in hex	Addressing mode	Argument length in bytes	Description
4051	Zero page	1	Memory = IMLI
4052	Zero page indexed	1	Memory = IMLI
8097	Absolute	2	Memory = IMLI (big endian)
8098	Absolute indexed	2	Memory = IMLI (as always big endian)
8099	Indirect	2	Memory = IMLI (as always big endian...)
809A	Indirect indexed	2	Memory = IMLI

STA – STore Accumulator

Memory = A

OPC in hex	Addressing mode	Argument length in bytes	Description
4026	Zero page	1	Memory = A
4027	Zero page indexed	1	Memory = A
8065	Absolute	2	Memory = A (as always big endian…)
8066	Absolute indexed	2	Memory = A
8067	Indirect	2	Memory = A
8068	Indirect indexed	2	Memory = A

STAH – STore Accumulator High

Memory = AH

OPC in hex	Addressing mode	Argument length	Description

		in bytes	
4028	Zero page	1	Memory = AH
4029	Zero page indexed	1	Memory = AH
8069	Absolute	2	Memory = AH (as always big endian…)
806A	Absolute indexed	2	Memory = AH
806B	Indirect	2	Memory = AH
806C	Indirect indexed	2	Memory = AH

STB – STore Base register

Memory = B

OPC in hex	Addressing mode	Argument length in bytes	Description
402A	Zero page	1	Memory = B
402B	Zero page indexed	1	Memory = B
806D	Absolute	2	Memory = B (as always big endian…)
806E	Absolute indexed	2	Memory = B
806F	Indirect	2	Memory = B
8070	Indirect indexed	2	Memory = B

STBH – STore Base register High

Memory = BH

OPC in hex	Addressing mode	Argument length in bytes	Description
402C	Zero page	1	Memory = BH
402D	Zero page indexed	1	Memory = BH
8071	Absolute	2	Memory = BH (as always big endian…)
8072	Absolute indexed	2	Memory = BH
8073	Indirect	2	Memory = BH
8074	Indirect indexed	2	Memory = BH

STI – STore Index register

Memory = I

OPC in hex	Addressing mode	Argument length in bytes	Description
402E	Zero page	1	Memory = A
8075	Absolute	2	Memory = A (as always big endian…)

8077	Indirect	2	Memory = A

STJ – STore "Jump to" register

Memory = J

OPC in hex	Addressing mode	Argument length in bytes	Description
402F	Zero page	1	Memory = J
8079	Absolute	2	Memory = J
807A	Absolute indexed	2	Memory = J
807B	Indirect	2	Memory = J
807C	Indirect indexed	2	Memory = J

STPC – STore Program Counter

Memory = PC

OPC in hex	Addressing mode	Argument length in bytes	Description
80A7	Absolute	2	Memory = PC

STSR – STore Status Register

Memory = SR

OPC in hex	Addressing mode	Argument length in bytes	Description
80A8	Absolute	2	Memory = SR

SUA – SUbtract Accumulator

A = A – operand

This operation may change the flags: N, O, Z, C.

OPC in hex	Addressing mode	Argument length in bytes	Description
0046	Base register	0	A = A – B
0048	Index register	0	A = A – I
4030	Immediate	1	AL = AL – argument (8-bit or byte)
4031	Zero page	1	A = A – word (16-bit) stored in the zero page
4032	Zero page indexed	1	A = A – word
807D	Absolute	2	A = A – word
807E	Absolute indexed	2	A = A – word

807F	Indirect	2	A = A – word
8080	Indirect indexed	2	A = A – word
809B	Immediate	2	A = A – argument (word)

SUB – SUbtract Base register

B = B – operand

This operation may change the flags: N, O, Z, C.

OPC in hex	Addressing mode	Argument length in bytes	Description
0047	Accumulator register	0	B = B – A
0049	Index register	0	B = B – I
4033	Immediate	1	BL = BL – argument (8-bit or byte)
4034	Zero page	1	B = B – word (16-bit) stored in the zero page
4035	Zero page indexed	1	B = B – word
8081	Absolute	2	B = B – word
8082	Absolute indexed	2	B = B – word
8083	Indirect	2	B = B – word
8084	Indirect indexed	2	B = B – word
809C	Immediate	2	B = B – argument (word)

SYS – SYStem call

Call to System software interrupt

OPC in hex	Addressing mode	Argument length in bytes	Description
3FFF	Implied	0	SYS interrupt is handled by a routine (go to page 4, memory mapping)

TAB – Transfer Accumulator to Base register

B = A

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
004A	Implied	0	B = A

TABH – Transfer Accumulator high to Base register High

BH = AH

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
002E	Implied	0	BH = AH

TABL – Transfer Accumulator low to Base register Low

BL = AL

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
005A	Implied	0	BL = AL

**TADR – Transfer Accumulator to Decremental Register*

DR = A

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
005B	Implied	0	DR = A

**TAEMH – Transfer Accumulator to EMHi register*

EMHI = A only if **System Privileges** is set.

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
005F	Implied	0	EMHI = A

**TAEML – Transfer Accumulator to EMLi register*

EMLI = A only if **System Privileges** is set.

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
0060	Implied	0	EMLI = A

TAHJ – Transfer Accumulator High to “Jump to” register

J = AH

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
0031	Implied	0	J = AH

TAI – Transfer Accumulator to Index register

I = A

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
004B	Implied	0	I = A

**TAIMH – Transfer Accumulator to IMHi register*

IMHI = A only if System Privileges is set.

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
005C	Implied	0	IMHI = A

**TAIML – Transfer Accumulator to IMLi register*

MLI = A only if System Privileges is set.

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
005D	Implied	0	IMLI = A

TBA – Transfer Base register to Accumulator

A = B

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
004C	Implied	0	A = B

TBAH – Transfer Base high to Accumulator High

AH = BH

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description

002D	Implied	0	AH = BH

TBAL – Transfer Base low to Accumulator Low

AL = BL

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
0059	Implied	0	AL = BL

TBHJ – Transfer Base High to "Jump to" register

J = BH

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
0032	Implied	0	J = BH

TBI – Transfer Base register to Index register

I = B

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
004D	Implied	0	I = B

TISP – Transfer Index register to Stack Pointer

SP = I

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
004E	Implied	0	SP = I

TSPB – Transfer Stack Pointer to Base register

B = SP

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
004F	Implied	0	B = SP

WRTE – WRiTE

Write into IO devices, data are stored in AH.

OPC in hex	Addressing mode	Argument length in bytes	Description
8085	Absolute	2	Write @ address argument, data are stored in AH
8087	Absolute indexed	2	Write @ address argument + I, data are stored in AH

WRTI – WRiTe using Index register as address

Write into IO devices, the address is stored in I.

OPC in hex	Addressing mode	Argument length in bytes	Description
0050	AH register	0	Write @ address stored in I, data are stored in AH
0051	BH register	0	Write @ address stored in I, data are stored in BH
4036	Immediate	1	Write @ address stored in I, data is the argument
4037	Zero page	1	Write @ address stored in I, data is a byte
8088	Absolute	2	Write @ address stored in I, data is a byte
8089	Indirect	2	Write @ address stored in I, data is a byte

XORA – eXclusive OR with Accumulator

A = A bit-wise ex or with operand

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
0052	Base register	0	A = A XOR B
0054	Index register	0	A = A XOR I
4039	Immediate	1	AL = AL XOR argument (byte)
403A	Zero page	1	A = A XOR word stored in the zero page
403B	Zero page indexed	1	A = A XOR word
809D	Absolute	2	A = A XOR word
809E	Absolute indexed	2	A = A XOR word
809F	Indirect	2	A = A XOR word
80A0	Indirect indexed	2	A = A XOR word

80A5	Immediate	2	A = A XOR argument (word)

XORB – eXclusive OR with Base register

B = B bit-wise ex or with operand

This operation may change the flags: N, Z.

OPC in hex	Addressing mode	Argument length in bytes	Description
0053	Accumulator register	0	B = B XOR A
0055	Index register	0	B = B XOR I
403C	Immediate	1	BL = BL XOR argument (byte)
403D	Zero page	1	B = B XOR word stored in the zero page
403E	Zero page indexed	1	B = B XOR word
80A1	Absolute	2	B = B XOR word
80A2	Absolute indexed	2	B = B XOR word
80A3	Indirect	2	B = B XOR word
80A4	Indirect indexed	2	B = B XOR word
80A6	Immediate	2	B = B XOR argument (word)