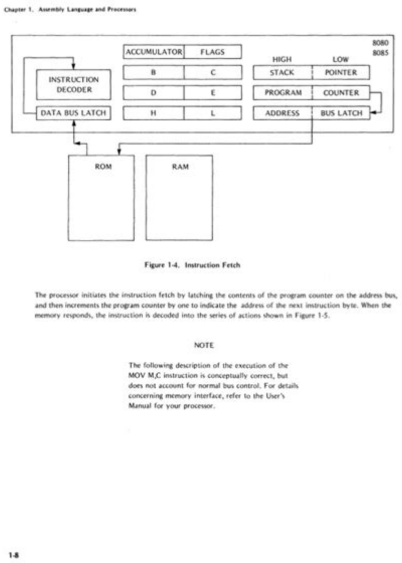
8085 assembly language programming manual



Chapter 1. Assembly Language and Processors

Figure 1-4. Instruction Fetch

The processor initiates the instruction fetch by latching the contents of the program counter on the address bus, and then increments the program counter by one to indicate the address of the next instruction byte. When the memory responds, the instruction is decoded into the series of actions shown in Figure 1-5.

NOTE

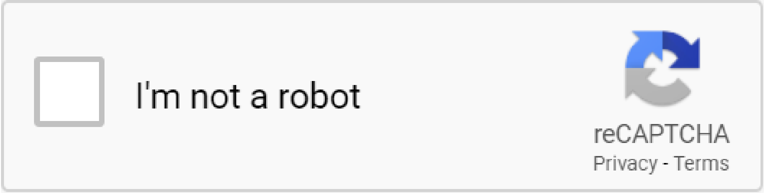
The following description of the execution of the MOV M,C instruction is conceptually correct, but does not account for normal bus control. For details concerning memory interface, refer to the User's Manual for your processor.

1-8

File Name: 8085 assembly language programming manual.pdf
Size: 2215 KB
Type: PDF, ePub, eBook
Category: Book
Uploaded: 28 May 2019, 13:42 PM
Rating: 4.6/5 from 656 votes.

Download Now!

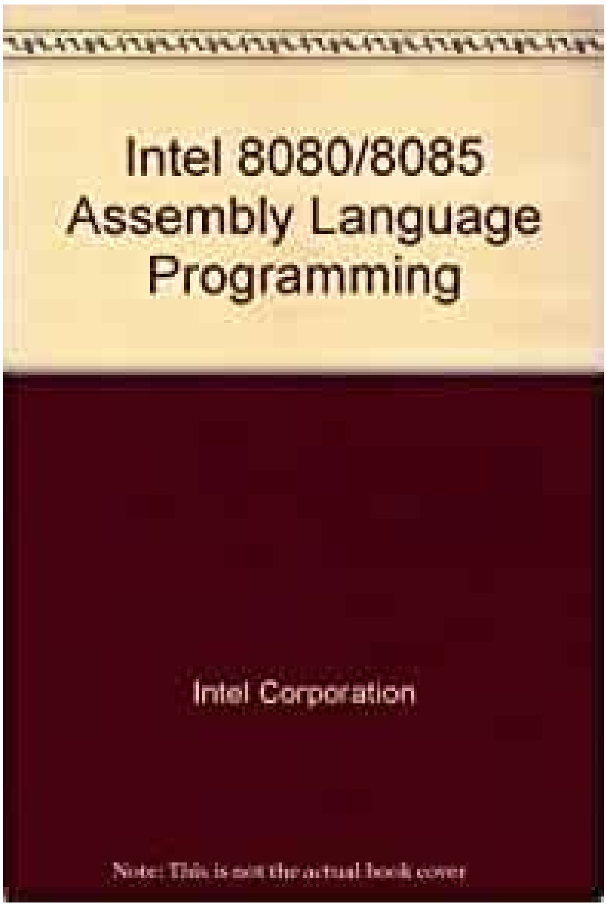
Please check the box below to proceed.



□ I'm not a robot

Book Descriptions:

8085 assembly language programming manual



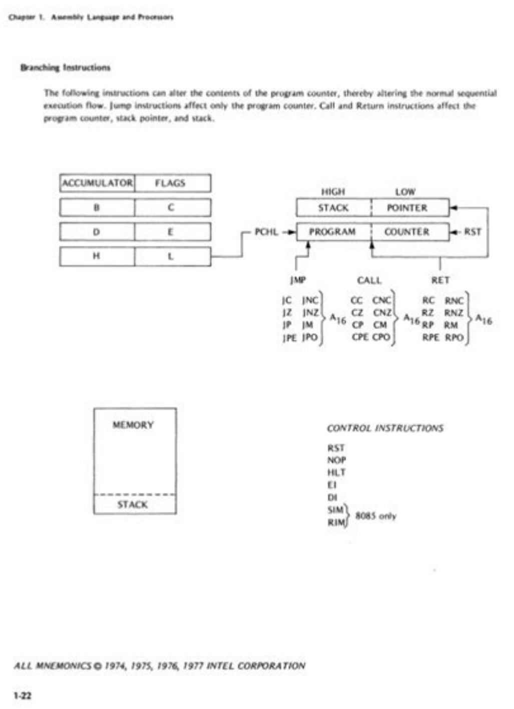
Intel 8080/8085 Assembly Language Programming

Intel Corporation

Note: This is not the actual book cover

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051 f. The information in this document is subject to change without notice. Intel Corporation mal To the programmer, the computer comprises, the following parts. Memory. The program counter. Work registers. Condition flags. The stack and stack pointer. The instruction set. Of the components listed above, memory is not part of the processor, but is of interest to the programmer. Since the program required to drive a microprocessor resides in memory, all microprocessor applications require Instructions and unchanging data With ROM you With RAM a program Notice, however, that storing programs Two special types of ROM PROM Programmable Read Only Memory and EPROM Eraseable Programmable. Read Only Memory are frequently used during program development. These memories are useful during In highvolume commercial Any time your program attempts to write any data to memory, that memory must be RAM. Also, if your pro If your program modifies any of its own instructions this procedure is The mix of ROM and RAM in an application is important to both the system designer and the programmer. Normally, the programmer must know the physical addresses of the RAM in the system so that data variables However, the relocation feature of this assembler allows you to code a The relocation Program Counter. With the program counter, we reach the first of the 8080s internal registers illustrated in Figure 13. The program counter keeps track of the next instruction byte to be fetched from memory which may be either. ROM or RAM. Each time it fetches an instruction byte from memory, the processor increments the program This process To alter the flow of program execution as The next instruction fetch occurs from the new address. Work Registers. The 8080 provides an 8bit accumulator and six other general purpose work registers, as shown in Figure 13.http://dongsuhk.com/userfiles/emco-pc-mill-55-manual.xml

- **8085 assembly language programming manual, 8085 assembly language programming manual youtube, 8085 assembly language programming manual pdf, 8085 assembly language programming manual download, 8085 assembly language programming manual free.**



Chapter 1. Assembly Language and Processors

Branching Instructions

The following instructions can alter the contents of the program counter, thereby altering the normal sequential execution flow. Jump instructions affect only the program counter. Call and Return instructions affect the program counter, stack pointer, and stack.

ACCUMULATOR
FLAGS
B
C
D
E
H
L
PCHL
HIGH
LOW
STACK
POINTER
PROGRAM
COUNTER
RST
JMP
CALL
RET
JC JNC
JZ JNZ
JP JM
JPE JPO
A_{16}
CC CNC
CZ CNZ
CP CM
CPE CPO
A_{16}
RC RNC
RZ RNZ
RP RM
RPE RPO
A_{16}
MEMORY
STACK

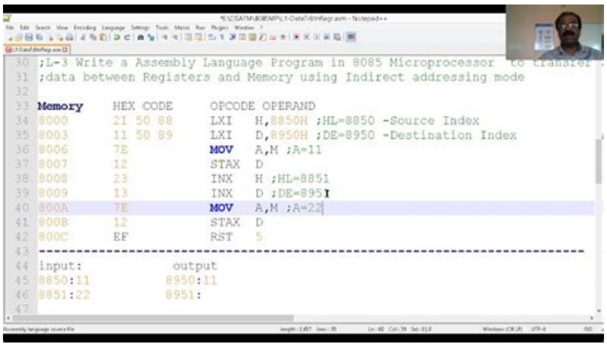
CONTROL INSTRUCTIONS

RST
NOP
HLT
EI
DI
SIM
RIM } 8085 only

ALL MNEMONICS © 1974, 1975, 1976, 1977 INTEL CORPORATION

1-22

Programs reference these registers by the letters A for the accumulator, B, C, D, E, H, and L. Thus, the Some instructions reference a pair of registers as shown in the following. Symbolic Reference Registers Referenced. B B and C. D D and E. H H and L. M H and L as a memory reference. PSW A and condition flags explained The symbolic reference for a single register is often the same as for a register pair. The instruction to be executed For example, ADD B is an 8bit operation. By contrast. PUSH B which pushes the contents of the B and C registers onto the stack is a 16bit operation. Notice that the letters H and M both refer to the H and L register pair. The choice of which to use depends on Use M when an instruction addresses memory via the H and L registers as in ADD. M add the contents of the memory location specified by the H and L registers to the contents of the accumu The general purpose registers B, C, D, E, H, and L can provide a wide variety of functions such as storing 8bit Because of A simple add to the accumulator, for example, can be accomplished by more than half a When possible, it is generally desirable to select a registertoregister instruction Also, using data already The accumulator also acts as a generalpurpose register, but it has some special capabilities not shared with the. Also, many operations involving the accumulator affect the condition flags as ex Example. The following figures illustrate the execution of a move instruction. The MOV M,C moves a copy of the contents Notice that this location must be in. RAM since data is to be written to memory. The processor initiates tiie instruction fetch by latching the contents of the program counter on the address bus, MOV M,C instruction is conceptually correct, but For details Manual for your processor. To execute the MOV M,C instruction, the processor latches the contents of the C register on the data bus and When the memory accepts the data, the processor Internal Work Registers.http://segtreinne.com.br/editor_imagens/emco-screen-door-manual.xml

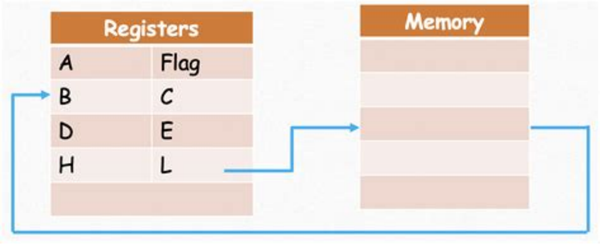


```
;L-3 Write a Assembly Language Program in 8085 Microprocessor  to transfer
;data between Registers and Memory using Indirect addressing mode

Memory     HEX CODE     OPCODE OPERAND
8000       21 50 88     LXI   H,8850H ;HL=8850 -Source Index
8003       11 50 89     LXI   D,8950H ;DE=8950 -Destination Index
8006       7E           MOV   A,M ;A=11
8007       12           STAX  D
8008       23           INX   H ;HL=8851
8009       13           INX   D ;DE=8951
800A       7E           MOV   A,M ;A=22
800B       12           STAX  D
800C       EF           RST   5
-------------------------------------------------------------------------------
input:            output
8850:11          8950:11
8851:22          8951:
```

Certain operations are destructive. For example, a compare is actually a subtract operation; a zero result indicates Since it is unacceptable to destroy either of the operands, the processor includes The programmer cannot access these registers. These registers are Condition Flags. The 8080 provides five flip flops used as condition flags. Certain arithmetic and logical instructions alter one or Your program can test the setting of four of these This allows you The fifth flag, auxiliary It is important for the programmer to know which flags are set by a particular instruction. Assume, for example, Coding a JPE jump if parity is even or JPG jump il parity is The jump executed by your program reflects the outcome of some previous For the operation to work correctly, you must include some instruc For example, you can This sets the parity flag without altering the data in the accumulator. In other cases, you will want to set a flag with one instruction, but then have a number of intervening instruc In these cases, you must be certain that the intervening instructions do not affect the The flags set by each instruction are detailed in the individual instruction descriptions in Chapter 3 of this Carry Flag. As its name implies, the carry flag is commonly used to indicate whether an addition causes a carry into the The carry flag is also used as a borrow flag in subtractions, as explained under Twos. Complement Representation of Data in Chapter 2 of this manual. The carry flag is also affected by the logical. AND, OR, and exclusive OR instructions. These instructions set ON or OFF particular bits of the accumulator. See the descriptions of the ANA, ANI, ORA, ORI, XRA, and XRI instructions in Chapter 3. The rotate instructions, which move the contents of the accumulator one position to the left or right, treat the See the descriptions of the RAL, RAR, RLC, and RRC Example. Addition of two onebyte numbers can produce a carry out of the highorder bit. Bit Number 7654 3210.

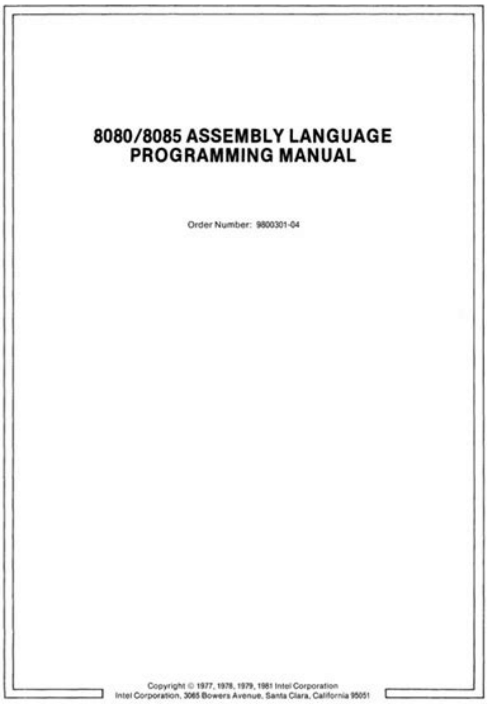
An addition that causes a carry out of the high order bit sets the carry flag to 1; an addition that does not cause Sign Flag. As explained under Twos Complement Representation of Data in Chapler 2, bit 7 of a result in the accumulator Instructions that affect the sign flag set the flag equal to bit 7. A zero in bit 7 This value is duplicated in the sign flag so that Zero Flag. Certain instructions set the zero flag to one to indicate that the result in the accumulator contains all zeros. These instructions reset the flag to zero if the result in the accumulator is other than zero. A result that has a Parity is determined by counting the number of one bits set in the result in the accumulator. Instructions that Auxiliary Carry Flag. The auxiliary carry flag indicates a carry out of bit 3 of the accumulator. You cannot test this flag directly in The auxiliary carry flag and the DAA instruction allow you to treat the value in the accumulator as two 4bit Thus, the value 0001 1001 is equivalent to 19. If this value is interpreted as a The DAA instruction requires the auxiliary carry flag since the BCD format makes it Chapter 1. Assembly Language and Processors. The auxiliary carry flag is affected by all add, subtract, increment, decrement, compare, and all logical AND. OR, and exclusive OR instructions. See the descriptions of these instructions in Chapter 3. There is some The 8080 logical AND Stack and Stack Pointer. To understand the purpose and effectiveness of the stack, it is useful to understand the concept of a subroutine. Assume that your program requires a multiplication routine. Since the 8080 has no multiply instructions, this You can recede this routine inline each time it is needed, Or, you can code a subroutine. Inline Coding. Subroutine When the call instruction is executed, the The contents of At the end of the subroutine, a Program execution then continues as though the subroutine had been coded inline.



http://afreecountry.com/?q=node/3018

The mechanism that makes this possible is, of course, the stack. The slack is simply an area of random access The slack pointer is a hardware register maintained by the processor. However, your program must initialize the stack pointer. This means that your program must load the base The base address of the stack is commonly assigned to the highest This is because the stack expands by decrementing the stack pointer. As items are As items are removed from the Nonetheless, the most recent item on the In terms of programming, a subroutine can call a subroutine, and so on. The only limitation to the number of items that can be added to the stack is the amount of RAM available for The amount of RAM allocated to the stack is important to the programmer. As you write your program, you For most applications, this To be more precise, Ultimately, your program should remove from Therefore, for any instruction that adds to the stack, you can sub The most critical factor is the maximum Otherwise, any Stack Operations. Stack operations transfer sixteen bits of data between memory and a pair of processor registers. The two basic A call instruction pushes the contents of the program counter which contains the address of the next instruction A return instruction pops sixteen bits off the stack and places them in the program counter. This requires the For example, if you call a subroutine and the subroutine The results are Saving Program Status. It is likely that a subroutine requires the use of one or more of the working registers. However, it is equally The subroutine can do this by Notice that PSW refers to The program status word is a 16bit word comprising the contents of the accumulator The IN and. OUT instructions initiate data transfers. The IN instruction latches the number of the desired port onto the address bus.

https://elitesoftsolutions.com/images/canon-e05-manual.pdf



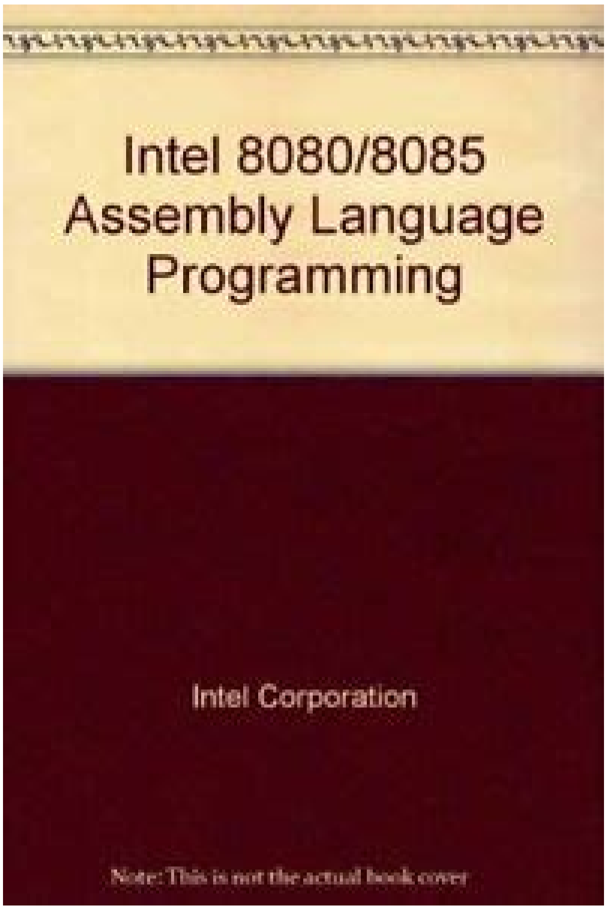
8080/8085 ASSEMBLY LANGUAGE
PROGRAMMING MANUAL

Order Number: 9800301-04

Copyright © 1977, 1978, 1979, 1981 Intel Corporation
Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

As soon as a byte of data is The OUT instruction latches the number of the desired port onto the address bus and latches the data in the The specified port number is duplicated on the address bus. Thus, the instruction IN 5 latches the bit configura Notice that the IN and OUT instructions simply initiate a data transfer. It is the responsibility of the peripheral Notice also that it is possible to dedicate any number of ports to You might use a number of ports as control signals, for example. Because input and output are almost totally application dependent, a discussion of design techniques is beyond For additional hardware information, refer to the 8080 or 8085 Microcomputer Systems Users Manual. For related programming information, see the descriptions of the IN, OUT, Dl, El, RST, and RIM and SIM Instruction Set. The 8080 incorporates a powerful array of instructions. This section provides a general overview of the instruc Addressing Modes. Implied Addressing. The addressing mode of certain instructions is implied by the instructions function. For Register Addressing. Quite a large set of instructions call for register addressing. With these instructions, you With these instructions, For example, the instruction CMP E may be interpreted as However, a few of these instructions For example, the PCHL instruction exchanges the contents of the program counter Immediate Addressing. Instructions that use immediate addressing have data assembled as a part of the instruction Hexadecimal 43 is the internal The processor fetches the next byte into one of its internal Notice that the names of the immediate instructions indicate that they use immediate data.

http://elipseradiologiadigital.com/images/canon-ef-manual-lens.pdf



Thus, the name of an All but two of the immediate instructions use the accumulator as an implied operand, as in the CPl instruction Thus, the instruction MVI D,OFFH moves the hexadecimal The LXI instruction load register pair immediate is even more unusual in that its immediate data is a 16bit As mentioned previously, your Direct Addressing. Jump instructions include a 16bit address as part of the instruction. For example, the Instructions that include a direct address require three bytes of storage one for the instruction code, and two Register Indirect Addressing. Register indirect instructions reference memory via a register pair. Thus, the Combined Addressing IVIodes. Some instructions use a combination of addressing modes. A CALL instruction, The direct address in a CALL instruction Timing Effects of Addressing Modes. Addressing modes affect both the amount of time required for executing For example, instructions that use implied or More important, however, is that the entire instruction can be fetched with a The number of memory accesses required is the single greatest factor in determining A CALL instruction, for example, requires The processor can access memory once during each processor cycle. Each cycle comprises a variable number of Thus, the timing of a four state instruction may range from Instruction Naming Conventions. The mnemonics assigned to the instructions are designed to indicate the function of the instruction. The instruc Data Transfer Group. The data transfer instructions move data between registers or between memory and MOV Move. MVI Move Immediate. LDA Load Accumulator Directly from Memory. STA Store Accumulator Directly in Memory. LHLD Load H and L Registers Directly from Memory. SHLD Store H and L Registers Directly in Memory An X in the name of a data transfer instruction implies that it deals with a register pair. LXI Load Register Pair with Immediate data. LDAX Load Accumulator from Address in Register Pair.

STAX Store Accumulator in Address in Register Pair. XCHG Exchange H and L with D and E. XTHL Exchange Top of Stack with H and L. Arithmetic Group. The arithmetic instructions add, subtract, increment, or decrement data in registers or ADD Add to Accumulator. ADI Add immediate Data to Accumulator. ADC Add to Accumulator Using Carry Flag. AC! Add Immediate Data to Accumulator Using Carry Flag. SUB Subtract from Accumulator. SUI Subtract Immediate Data from Accumulator.

SBB Subtract from Accumulator Using Borrow Carry Flag. SBI Subtract Immediate from Accumulator Using Borrow. INR Increment Specified Byte by One. DCR Decrement Specified Byte by One. INX Increment Register Pair by One. DCX Decrement Register Pair by One. DAD Double Register Add Add Contents of Register. Pair to H and L Register Pair. Logical Group. This group performs logical Boolean operations on data in registers and memory and on The logical AND, OR, and Exclusive OR instructions enable you to set specific bits in the accumulator ON or ANI Logical AND with Accumulator Using Immediate Data. ORA Logical OR with Accumulator. ORI Logical OR with Accumulator Using Immediate Data. XRA Exclusive Logical OR with Accumulator. XRl Exclusive OR Using Immediate Data. The compare instructions compare the contents of an 8bit value with the contents of the accumulator. CMP Compare. CPI Compare Using Immediate Data The rotate instructions sliift the contenls of the accumulator one bit position to the left or right. RLC Rotate Accumulator Left. RRC Rotate Accumulator Right. RAL Rotate Left Through Carry. RAR Rotate R ght Through Carry. Complement and carry flag instructions. CMA Complement Accumulator. CMC Complement Carry Flag. STC Set Carry Flag. Branch Group. The branching instructions alter normal sequential program flow, either unconditionally or CALL Call. RET Return.

http://www.kocay.com.tr/wp-content/plugins/formcraft/file-upload/server/content/files/16274c3b364d90---brother-hl-1030-manual.pdf

Conditional branching instructions examine the status of one of four condition flags to determine whether the The conditions that may be specified are as follows. Tiius, the conditional branching instructions are specified as follows. Jumps Calls Returns. JC CC RC Carry. JNC CNC RNC No Carry. JZ CZ RZ Zero. JNZ CNZ RNZ Not Zero Two other instructions can effect a branch by replacing the contents of the program counter. PCHL Move H and L to Program Counter. RST Special Fiestart Instruction Used with Interrupts PUSH Push Two Bytes of Data onto the Stack. POP Pop Two Bytes of Data off the Stack. SPHL Move contents of H and L to Stack Pointer. IN Initiate Input Operation. OUT Initiate Output Operation. The machine control instructions are as follows. E\ Enable Interrupt System. Dl Disable Interrupt System. HLT Halt. NOP No Operation. The following illustrations graphically summarize the instruction set by showing the hardware acted upon by The type of operand allowed for each instruction is indicated through the use of a code. When no code is given, the instruction does not allow operands. Code Meaning. REGMo The operand may specify one of the 8bit registers A,B,C,D,E,H, or L or M MOV instruction, which calls for two operands, can specify M for only one Do Designates 8bit immediate operand. A,r Designates a 16bit address. Po Designates an 8bit port number. SBl. ANl OUT Pg Register Pair Word Instructions. The following instructions all deal with 16bit words. DAD affects only the carry Branching Instructions. The following instructions can alter the conent5 of the program counter, thereby altering the normal sequential Call and Return instructions affect the Dl Chapter 1. Assembly Language and Processors. The following is a summary of the instruction set DCRf MOV REGMg.REGMg Dl The MOV instruction, which Designates 8bit immediate operand. Designates a 16bit address. Designates an 8bit port number. Designates a 16 bit immediate operand.

autoescuelatosal.com/galeria/files/canon-sx20is-instruction-manual.pdf

Except for two additional instructions, the 8085 instruction set is identical to and fully Most programs written for the 8080 should operate on the 8085 with A partial listing of 8085 design features includes the following. Execution speeds approximately 50% faster than the 8080. Incorporation in the processor of the features of the 8224 Clock Generator and Driver and the A nonmaskable TRAP interrupt for handling serious problems such as power failures. Three separately maskable interrupts that generate internal RST instructions. Programming for the 8085. For the programmer, the new features of the 8085 are summarized in the two new instructions SIM and RIM. These instructions differ from the 8080 instrjctions in that each has multiple functions. The SiM instruction. The programmer must place the desired interrupt The RIM instruction Details of these instruc Despite the new interrupt features of the 8085, programming for interrupts is little changed. Notice, however, that Therefore, Also, the TRAP inlerrupt input is nonmaskable and The interrupts have the following priority. TRAP highest When more than one interrupt is pending, the processor always recognizes the higher priority interrupt first. These priorities apply only to the sequence in which interrupts are recognized. Program routines that service Thus, an RST5.5 interrupt can interrupt the service routine for an RST7.5 Conditional Instructions. Execution of conditional instructions on the 8085 differs from the 8080. The 8080 fetches ail three instruction The 8085 evaluates the condition while it fetches the second Skipping the unnecessary byte allows for faster execution. The source line This assembler recognizes three types of source lines; instructions, directives, and controls. This manual describes Controls are described in the operators manual for your version of the assembler. This chapter describes the general rules for coding source lines.

Specific instructions see Chapter 3 and Even so, the coding of such instructions and Label\ Opcode Operand;Comment. Name. The fields may be separated by any number of blanks, but must be separated by at least one delimiter. Each Character Set. The following characters are legal in assembly language source statements. Internally, Character Meaning. Minus sign Labels are always optional. An instruction label is a symbol name whose value is the location where the instruc A symbol used Alphanumeric characters include the letters of the alphabet, the question mark character, and the decimal A name is required for the SET, EQU, and MACRO directives. Names follow the same coding rules as labels, Opcode Field Operand Field. The operand field identifies the data to be operated on by the specified opcode. Some instructions require no As a general rule, when two operands are required as in data Examples. MOV A,C;iV10VE CONTENTS OF REG C TO ACCUMULATOR The optional comment field may contain anv information you deem useful for annotating your program. The Because the semicolon is a delimiter, However, spaces are Although comments arc always optional, you should Hexadecimal Data. Each hexadecimal number must begin with a numeric digit 0 through 9 and must be Label Opcode Operand Comment Thus, the following statements are equivalent. Label Opcode Operand Comment Label Opcode Operand Comment Label Opcode Operand Comment The location counter contains the Label Opcode Operand Comment Label Comment The SET and EQU directives can assign values to labels. In the iollowing example, Label. Opcode. Operand. Comment. Al The label assigned to an instruction or a data definition has as its value the Instructions elsewhere in the program can refer to this Label Because the rules for coding expressions are rather extensive, further discussion of expressions is deferred until Instructions as Operands.

One operand type was intentionally omitted from the list of operand field infor The operand has the value of Label Opcode Operand RegisterType Operands. Only instructions that allow registers as operands may have registertype operands. Expressions containing registertype operands are flagged as errors. Thus, an instruction like The only assembler directives that may contain registertype operands are EQU, SET, and actual parameters in Any particular combination may For example, the code IFH may be interpreted as an instruction Rotate. Accumulator Right Through Carry, as the hexadecimal value IF, the decimal value 31, or simply the bit Arithmetic instructions assume that the data bytes upon which they operate are in the twos complement To form the tens complement of a The ability to perform subtraction with a form of addition is a great advantage in a computer since fewer cir The processor forms the twos complement of a binary value simply by reversing the value of each bit and then Any carry out of the high order bit is ignored when the complement is formed. Thus, Again, by disregarding the carry out of the high order position, the subtraction is performed through a form of This is because the processors complement the carry flag at the end of a subtract In the example shown, no borrow By contrast, the carry flag is set ON if we subtract 35 from 12 Therefore, the processor sets the carry flag ON. Notice also that the result is stored in a complemented form. If you want to interpret this result as a decimal value, you must again form its twos complement; When a byte is interpreted as a signed twos complement number, A zero in this bit indicates a positive number, a one a negative number. The At the beginning of this description of twos complement arithmetic, it was stated that any 8bit byte may con It must also be stated that the proper interpretation As an example, consider the compare instruction.

The compare logic considers only the raw bit values of the Therefore, a negative twos complement number always compares higher than a positive As a result, the meanings of the flags set by Your program must account for this condition. The locations in program memory can be compared to a cluster of post office boxes. Suppose Richard Roe He can then ask for his letters by saying Give me the mail in box 500, or The content of the post office box can be accessed by a fixed, absolute address The postal clerk correlates the symbolic names and their absolute values The assembler references its It tells the assembler where the next Symbol Characteristics. A symbol can contain one to six alphabetic AZ or numeric 09 characters with the first character alphabetic Symbols of the The assembler regards symbols as having the following attributes reserved or userdefined; global or limited; Reserved, UserDefined, and AssemblerGenerated Symbols. Reserved symbols are those that already have special meaning to the assembler and therefore cannot appear as The mnemonic names for nnachine instructions and the assembler directives are all reserved The following instruction operand symbols are also reserved. Symbol. Meaning Accumulator register. Register B or register pair B and C. Register C. Register D or register pair D and E. Register E. Register H or register pair H and L. Register L. Stack pointer register. Program status word Contents of A and status flags. Memory reference code using address in H and L. Special relocatability feature. Special relocatability feature Userdefined symbols are symbols you create to reference instruction and data addresses. These symbols are Assemblergenerated symbols are created by the assembler to replace userdefined symbols whose scope is limited Global and Limited Symbols. Most symbols are global. This means that they have meaning throughout your program.

Assume, for example, You may then code a jump or a call to RTN from any If you assign the symbolic name RTN to a second routine, an error results since you Certain symbols have meaning only within a macro definition or within a call to that macro; these symbols are Macros require local symbols because the same macro may be used many times in the See Chapter 5 for additional information about macros. Permanent and Redefinable Symbols. Most symbols arc permanenl since their value cannot change during the assembly operation. Only symbols Absolute and Relocatable Symbols. An important attribute of symbols with this assembler is that of relocatability. Relocatable programs are. These programs are later relocated to some other set of memory Symbols with This distinction becomes important when External and public symbols are special types of relocatable symbols. These symbols are required to establish Such symbols must appear in an EXTRN statetiient, or the assembler will flag them as undefined. Conversely, PUBLIC symbols are defined in the current program module, but may be accessed by other Absolute and relocatable symbols may both appear in a relocatable module. References to any of the assembler But these references arc valid in any module. Each element of an expression is a term. Expressions, like symbols, may be absolute or relocatable. For the sake of readers who do not require the However, users of relocation should read all the Operators. The assembler includes five groups of operators which permit the following assemblylime operations; arithmetic It is important Once the assembler has evaluated an expression, it Assume, for example, that your program defines a list of ten con Arithmetic Operators. The arithmetic operators are as follows. Operator Unary or binary addition. Unary or binary subtraction. Multiplication. Division by zero causes an error. Modulo. Result is the remainder caused by a. Examples. Notice that the MOD operator must be separa.

ted from its operands by spaces Siiift Operators. Operator Meaning. Shift operand y to the right x bit positions. Shift operand y to the left x bit positions. The shift operators do not wraparound any tiits shifted out of the byte. Bit positions vacated by the shift Notice that the shift operator must be separated from its operands by spaces. Example. Assume that NUMBR has the value 0101 0101. The effects of the shift operators is as follows Notice that a shift one bit position to the left has the effect of multiplying a value by two; a shift one bit Logical Operators. The logical operators are as follows. Operator Logical ones complement. The logical operators act only upon the least significant bit of values involved in the operation. Also, these These directives are fully explained in Chapter 4. The following IF directive tests the least significant bit of three items. The assembly language code that follows This means that all three fields must have a one bit in the The compare operators are as follows. Operator Equal. Not equal. Less than. Less than or equal. Greater than. Greater than or equal. Special operator used to test for null missing macro The compare operators yield a yesno result. Thus, if the evaluation of the relation is TRUE, the value of the Relational operations are based strictly on magni Thus, a twos complement negative number which always has a one in its high Since the NUL operator applies only to the nacro feature, NUL is described in Chapter 5. The compare operators are commonly used in conditional IF directives. These directives are fully explained in. Chapter 4. Notice that the compare operator must be separated from its operands by spaces. The following IF directive tests the values of FLDl and FLD2 for equality. If the result of the comparison is. TRUE, the assembly language coding following the IF directive is assembled. Otherwise, the code is skipped over. IF FLDl EQ FLD2. Byle Isolation Operators.