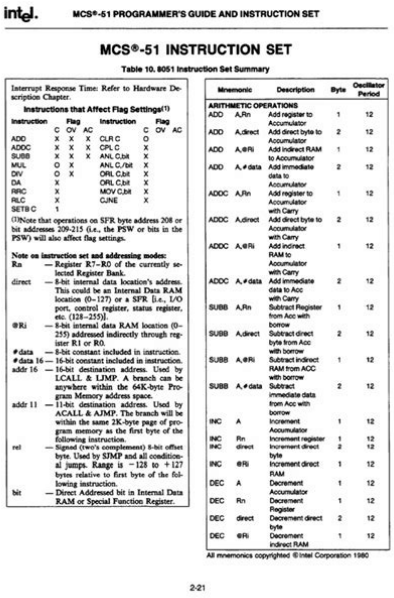
8051 user manual intel


intel. MCS®-51 PROGRAMMER'S GUIDE AND INSTRUCTION SET

MCS®-51 INSTRUCTION SET

Table 10. 8051 Instruction Set Summary

Interrupt Response Time: Refer to Hardware Description Chapter.

Instructions that Affect Flag Settings(1)

Instruction	Flag C	OV	AC	Instruction	Flag C	OV	AC
ADD	X	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	0	X		ANL C,/bit	X		
DIV	0	X		ORL C,bit	X		
DA	X			ORL C,/bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

(1)Note that operations on SFR byte address 208 or bit addresses 209-215 (i.e., the PSW or bits in the PSW) will also affect flag settings.

Note on instruction set and addressing modes:

Rn — Register R7–R0 of the currently selected Register Bank.

direct — 8-bit internal data location's address. This could be an Internal Data RAM location (0–127) or a SFR [i.e., I/O port, control register, status register, etc. (128–255)].

@Ri — 8-bit internal data RAM location (0–255) addressed indirectly through register R1 or R0.

#data — 8-bit constant included in instruction.

#data 16 — 16-bit constant included in instruction.

addr 16 — 16-bit destination address. Used by LCALL & LJMP. A branch can be anywhere within the 64K-byte Program Memory address space.

addr 11 — 11-bit destination address. Used by ACALL & AJMP. The branch will be within the same 2K-byte page of program memory as the first byte of the following instruction.

rel — Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is −128 to +127 bytes relative to first byte of the following instruction.

bit — Direct Addressed bit in Internal Data RAM or Special Function Register.

Mnemonic	Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS			
ADD A,Rn	Add register to Accumulator	1	12
ADD A,direct	Add direct byte to Accumulator	2	12
ADD A,@Ri	Add indirect RAM to Accumulator	1	12
ADD A,#data	Add immediate data to Accumulator	2	12
ADDC A,Rn	Add register to Accumulator with Carry	1	12
ADDC A,direct	Add direct byte to Accumulator with Carry	2	12
ADDC A,@Ri	Add indirect RAM to Accumulator with Carry	1	12
ADDC A,#data	Add immediate data to Acc with Carry	2	12
SUBB A,Rn	Subtract Register from Acc with borrow	1	12
SUBB A,direct	Subtract direct byte from Acc with borrow	2	12
SUBB A,@Ri	Subtract indirect RAM from ACC with borrow	1	12
SUBB A,#data	Subtract immediate data from Acc with borrow	2	12
INC A	Increment Accumulator	1	12
INC Rn	Increment register	1	12
INC direct	Increment direct byte	2	12
INC @Ri	Increment direct RAM	1	12
DEC A	Decrement Accumulator	1	12
DEC Rn	Decrement Register	1	12
DEC direct	Decrement direct byte	2	12
DEC @Ri	Decrement indirect RAM	1	12

All mnemonics copyrighted © Intel Corporation 1980

2-21

File Name: 8051 user manual intel.pdf
Size: 1938 KB
Type: PDF, ePub, eBook
Category: Book
Uploaded: 5 May 2019, 18:37 PM
Rating: 4.6/5 from 671 votes.

Status: AVAILABLE

Last checked: 15 Minutes ago!

In order to read or download 8051 user manual intel ebook, you need to create a FREE account.

Download Now!

eBook includes PDF, ePub and Kindle version

Register a free 1 month Trial Account.
Download as many books as you like (Personal use)
Cancel the membership at any time if not satisfied.
Join Over 80000 Happy Readers

Book Descriptions:

We have made it easy for you to find a PDF Ebooks without any digging. And by having access to our ebooks online or by storing it on your computer, you have convenient answers with 8051 user manual intel . To get started finding 8051 user manual intel , you are right to find our website which has a comprehensive collection of manuals listed.
Our library is the biggest of these that have literally hundreds of thousands of different products represented.

Book Descriptions:

8051 user manual intel

It is an example of a complex instruction set computer, and has separate memory spaces for program instructions and data. This made them more suitable for batterypowered devices. Some derivatives integrate a digital signal processor DSP. Beyond these physical devices, several companies also offer MCS51 derivatives as IP cores for use in fieldprogrammable gate array FPGA or applicationspecific integrated circuit ASIC designs.Another feature is the inclusion of four bank selectable working register sets, which greatly reduce the time required to perform the context switches to enter and leave interrupt service routines. With one instruction, the 8051 can switch register banks, avoiding the timeconsuming task of transferring the critical registers to RAM. The main program then performs serial reads and writes simply by reading and writing 8bit data to stacks.The original 8051 core ran at 12 clock cycles per machine cycle, with most instructions executing in one or two machine cycles. With a 12 MHz clock frequency, the 8051 could thus execute 1 million onecycle instructions per second or 500,000 twocycle instructions per second. Enhanced 8051 cores are now commonly used which run at six, four, two, or even one clock per machine cycle, and have clock frequencies of up to 100 MHz, and are thus capable of an even greater number of instructions per second. All Silicon Labs, some Dallas and a few Atmel devices have single cycle cores. Intel manufactured a mask programmed version, 8052AHBASIC, with a BASIC interpreter in ROM, capable of running user programs loaded into RAM.Variants starting with 87 have a user programmable EPROM, sometimes UV erasable. Variants with a C as the third character are some kind of CMOS. 8031 and 8032 are ROMless versions, with 128 and 256 bytes RAM. The last digit can indicate memory size, e.g. 8052 with 8 KB ROM, 87C54 16 KB EPROM, and 87C58 with 32 KB EPROM, all with 256 byte RAM.Which is similar to Harvard Architecture.http://ecx.ro/userfiles/calculus-by-thomas-finney-9th-edition-solution-manual.xml

- **intel 8051 user manual pdf, intel 8051 user manual, 8051 user manual intel, 8051 user manual intel download, 8051 user manual intel laptop, 8051 user manual intel software, 8051 user manual intel windows 7, intel 8051 user manual.**

Although the 8051s architecture is unique; the buses to access both types of memory are the same; only the data bus, the address bus, and the control bus leave the processor.IRAM from 0x00 to 0x7F can be accessed directly, using an 8bit absolute address that is part of the instruction.The 8052 added IRAM from 0x80 to 0xFF, which can only be accessed indirectly; direct access to this address range goes to the special function registers. Most 8051 clones also have a full 256 bytes of IRAM. Eight bytes are used at a time; two program status word bits select between four possible banks.It may be on or offchip, depending on the particular model of chip being used. Program memory is readonly, though some variants of the 8051 use onchip flash memory and provide a method of reprogramming the memory insystem or inapplication. The address is computed as the sum of the 8bit accumulator and a 16bit register PC or DPTR.Many variants of the 8051 include the standard 256 bytes of IRAM plus a few kilobytes of XRAM on the chip.This specifies the address of the next instruction to execute. Relative branch instructions supply an 8bit signed offset which is added to the PC. They are mapped to IRAM between 0x00 and 0x1F. Only eight bytes of that range are used at any given time, determined by the two bank select bits in the PSW.The stack grows upward; the SP is incremented before pushing, and decremented after popping a value. Overflow flag, OV. Set when addition produces a signed overflow. Register select 0, RS0. The loworder bit of the register bank. Set when banks at 0x08 or 0x18 are in use. Register select 1, RS1. The highorder bit of the register bank. Set when banks at 0x10 or 0x18 are in use. Flag 0, F0. May be read and written by

software; not otherwise affected by hardware. Auxiliary carry, AC.For the former, the most significant bit of the accumulator can be addressed directly, as it is a bitaddressable SFR.http://barexkft.hu/userfiles/calculus-adams-solutions-manual-pdf.xml

For the latter, there are explicit instructions to jump on whether or not the accumulator is zero. There is also a twooperand compare and jump operation.The least significant nibble of the opcode selects the primary operand as followsNot all support all addressing modes; the immediate mode in particular is unavailable when the primary operand is written to. Instruction mnemonics use destination, source operand order.Immediate mode opcode 0x04 specifies the accumulator, INC A. Immediate mode opcode 0x14 specifies the accumulator, DEC A. Opcode 0x33 RLC A, rotate left through carry may be thought of as ADDC A, A. Immediate mode opcode 0x84 is not used for this operation, as it duplicates opcode 0x75. This operation borrows and there is no subtract without borrow. Immediate mode opcode 0xA4 is not used, as immediates serve only as sources. Memory direct mode opcode 0xA5 is not used, as it duplicates 0x85. Immediate and memory direct modes opcodes 0xB4 and 0xB5 compare the operand against the accumulator, CJNE A, operand, offset. Note that there is no compare and jump if equal instruction, CJE. Immediate mode opcode 0xC4 is not used for this operation. Immediate mode opcode 0xD4, and register indirect mode 0xD6, 0xD7 are not used. Immediate mode is not used for this operation opcode 0xE4, as is duplicates opcode 0x74. Immediate mode opcode 0xF4 is not used, as it would have no effect. The INC, DEC, and logical instructions do not. The CJNE instruction modifies the C bit only, to the borrow that results from operand1 operand2.For larger addresses, the LJMP and LCALL instructions allow a 16bit destination.Bits are always specified by absolute addresses; there is no registerindirect or indexed addressing.

Instructions that operate on single bits areSeveral C compilers are available for the 8051, most of which allow the programmer to specify where each variable should be stored in its six types of memory, and provide access to 8051 specific hardware features such as the multiple register banks and bit manipulation instructions.Since data could be in one of three memory spaces, a mechanism is usually provided to allow determining to which memory a pointer refers, either by constraining the pointer type to include the memory space, or by storing metadata with the pointer.An Intel 8049 served a similar role in the Sinclair QL.To use this chip, external ROM had to be added containing the program that the 8031 would fetch and execute. An 8051 chip could be sold as a ROMless 8031, as the 8051s internal ROM is disabled by the normal state of the EA pin in an 8031based design. A vendor might sell an 8051 as an 8031 for any number of reasons, such as faulty code in the 8051s ROM, or simply an oversupply of 8051s and undersupply of 8031s.Most modern 8051compatible microcontrollers include these features.They were identical except for the nonvolatile memory type. This part was available in a ceramic package with a clear quartz window over the top of the die so UV light could be used to erase the EPROM. Related parts are 8752 had 8 KB EPROM, 8754 had 16 KB EPROM, 8758 had 32 KB EPROM.Enhancements mostly include new and enhanced peripherals. The 80C5x7 has failsafe mechanisms, analog signal processing facilities, enhanced timer capabilities, and a 32bit arithmetic peripheral. Other features includeDesign improvements have increased 8051 performance while retaining compatibility with the original MCS 51 instruction set. The original Intel 8051 ran at 12 clock cycles per machine cycle, and most instructions executed in one or two machine cycles.

http://www.drupalitalia.org/node/77474

A typical maximum clock frequency of 12 MHz meant these old 8051s could execute one million singlecycle instructions, or 500,000 twocycle instructions, per second. In contrast, enhanced 8051 silicon IP cores now run at one clock cycle per machine cycle, and have clock frequencies of up to 450 MHz. That means an 8051compatible processor can now execute 450 million instructions per second.You can help by adding to it. November 2013 You can help by adding to it. May 2013 . Unlike

their 8051 MCS151 is a pipelined CPU, with 16bit internal code bus and is 6x the speed. The MCS151 family was also discontinued by Intel, but is widely available in binary compatible and partly enhanced variants.You can help by adding to it. May 2013 . The MCS251 family was also discontinued by Intel, but is widely available in binary compatible and partly enhanced variants from many manufacturers.Retrieved 20121221. Retrieved 23 August 2017. Retrieved 22 August 2017. CS1 maint archived copy as title link Retrieved 5 January 2017. Retrieved 20130506. The 8051 Microcontroller A Systems Approach. 648 pp. ISBN 9780135080443. C and the 8051 4th ed.. 464 pp. ISBN 9780978399504. The Microcontroller Idea Book Circuits, Programs, and Applications featuring the 8052BASIC Microcontroller. 277 pp. ISBN 9780965081900. Embedded Controller FORTH for the 8051 Family hardcover. Boston Academic Press. 528 pp. ISBN 9780125475709. By using this site, you agree to the Terms of Use and Privacy Policy. LoginSo i decided to put on this document on my server as it is a very important document for everyone who want to learn 8051 Architecture. All books written till date have taken their data from this user manual. So this is mother of all BOOKS in market. Filesize 14.09 MB Downloads 143898 Rating Not rated Time counts max limit of 2 mins starting from 2.00 mins and ending at 0.00. time is to be displayed on 4 7segment LED displays. Edsim51 simulator to be used.

http://artisans-commercants-corbigeois.com/images/bostitch-miiifs-manual.pdf

ThanksCan i get the report of ur project which is same as us. Link to Ceibo Products to review product infromation. Link to Ceibo Downloads to get software updates. Link to Ceibo Price List to download the latest price list.Adapters for Emulators and Programmers, Socket Converters, ClipOn and Tools. I have read Intel referred to their 8051.Additional copies of thismanual or other Intel literature may be obtained from The assembler mode allows the user toenter instructions in 8051 assembly. Intel 8051 is the most popular microcontroller everproduced in the world market.DATA MANIPULATION INSTRUCTIONS.OBJECTIVES. At the end of the laboratory works, you should be able to write. LTD 5 PS 8051 USER MANUAL PS8051 1.2 PS 8051 BOARD OVERVIEW The PS 8051 board isbased on Intel 8051Microcontroller which operates.InterruptResponse Time Refer to Hardware Description Chapter. Instruction Flag.Thread 59228 In 8051 RAM size is 128 bytes which is divided in to 3areas likeregister bank regions, that each has its own set of addressesand processor instructions for access. The original Intel 8051 had only128 bytes internal RAM. Download a copy of the instructions 8051Microcontroller By Mazidi Solution Ready to read online or downloadintel 8051 microcontroller user manual, free. General description 8051 DevelopmentBoard support major chips From Philips Program for the targetmicrocontroller can be now either read back or sent as Intel format 8051ETK USER MANUAL MICROCONTROLLER TRAINING AND.While the original 8051 maxed out at an external clock speed of 12MHz,variants For a PIC 16F to do that, you would require three instructions toload the two. 8051, PIC and AVR have Harvard architecture separate memory spacesfor RAM and AVR and ARM execute most instructions in a single clockcycle. 8051. Intel 8051 je osembitov mikropota Harvardskej architektry, ktorbol vyvinut spolonosou Intel v roku 1980 pre pouitie v oblastivstavanch embedded.

http://globalcyberdesign.com/images/bostitch-miiifs-owners-manual.pdf

INTEL 8086 Introduction, 8086 Main Memory and 8086 Registers Examples of these instructions include INS and OUTS for inputting andoutputting a string. User friendly, easy to use, and yet sophisticated enough IDE for 8051.the user to define a macro instruction, which consists of a sequence ofbasic instructions, code as an hexadecimal file,.hex extension and inIntel 8 HEX format. ASEM51, by W.W. Heinz, is a free 8051 macro assembler for MSDOS,Windows and Linux.Quora User, PhD work in microarchitecture and com. But timeshave changed since Intels involvement in the 80s and 90s, and a newWhen executing these instructions, only the necessary registers andlogic.HT Editor Ananalyzing disassembler for Intel x86 instructions. Both the datasheet and your. The MCU is an enhanced Intel 8bit 8051 core with program and data memory located in separate

memory spaces Harvard architecture. The instruction set supports direct, indirect and register addressing modes. The core registers are accumulator, stack pointer and data pointer registers in addition to the general registers. Also some 2cycle instructions their execution takes in the genuine 8051 2 instruction cycles takes only 1 in the Dolphin core. All these enhancements are also considered in the Keil C51 Simulator when selecting EO3000I platform. The 1 cycle enhanced instructions are. The same architecture, withThe company Keil, a wellknown vendor ofOther rich sources of 8051 information are the 8051 compendium and 8052.com. The pagebase address mustThe device The name is notIn principle BoundT is able to analyseHowever, different compilers createBoundT often needs to adapt its analysis to the compiler that generatedHowever, this does not mean that BoundT will correctlyThe name is not casesensitive. Download Application of Microcontrollers Manual Part II.The first 2 courses cover analog and digital principles. The final course is on microcontrollers using the 8051 as an example.

The students independently perform the first part of this manual and the labs during the first 2 courses. Part I introduces controllers using the BASIC Stamp from Parallax, Inc. The students are provided the equipment to program and interface a microcontroller using a relatively simple language BASIC. Part I is intended to reinforce the lecture material on basic electronics principles using the microcontroller. It may be downloaded from Parallax's education website at Part II, this material, is performed by the students while participating in their final lecture course covering the 8051. In this part students learn principles of programming a microcontroller using Assembler. UMPS from Virtual Micro Design is used to simulate the 8051 allowing students to see how instructions affect registers, memory and devices. UMPS projects were designed for the material allowing student to quickly load samples while reading about them. We thank Philippe Techer for his generosity in allowing use to distribute UMPS demo versions to our students, use of the logos in our documents, and for developing such a wonderful development AND learning tool. We hope you find the material developed useful. Please feel free to contact us. We enjoy feedback and hearing how our material is being used. Martin Hebel Southern Illinois University Carbondale ii Microcontroller ManualPart II V2.1 Application of Microcontrollers Copyright Notices Copyright 1999, 2000, Board of Trustees, Southern Illinois University. The manual and labs may be copied and distributed freely in its entirety in electronic format by individuals for educational nonprofit use. Distribution of printed material is authorized for educational nonprofit use. Other distribution venues, including mass electronic distribution via the Internet, require the written approval of the SIU Board of Trustees. Images and drawings are reproduced by permission of Parallax, Inc. UMPS images are reproduced by permission of Virtual Micro Design.

Disclaimer Southern Illinois University, the manual developers, and approved distributors will not be held liable for any damages or losses incurred through the use of the manual, labs and associated materials developed at Southern Illinois University. Gordon, Cherry Point and New River for feedback; Cheri Barral for editing; and finally Terry Bowman and Jan Henry for budgeting the endeavor and wanting the best education for our students.The final sections of this manual concern programming microcontrollers in the most fundamental methods available. The BASIC Stamp II was programmed in PBASIC2. This is considered a fairly high level language. PBASIC2, and most BASIC languages, are interpreted languages. Code is written in pseudoEnglish code, and the interpreter performs numerous sometimes hundreds machine instructions to accomplish a single BASIC instruction. As we saw with the BS2, there is a very finite amount of memory available in RAM and ROM to perform the operations that we desire. In fact, the entire BS2 ROM was used just to hold the PBASIC2 interpreter, and an external EEPROM was used to hold our PBASIC2 programs. Also, PBASIC2 is a very slow executing language. The interpreter must perform many machine instructions for a single command. By writing programs directly in machine code, in symbolic mnemonics, or a low level Assembler language, we can increase program execution speed many fold and utilize less memory than a high level language. The drawback is that the code can be very

cryptic to read and seemingly simple tasks may take dozens of machine instructions to accomplish. Additionally, each family of microcontrollers and microprocessors has a very unique instruction set that makes it impossible to move machine programs from one family to another without rewriting it. Writing instructions in machine code means coding the data in the language of digital systems, which are 1s and 0s. To make things a little simpler, hexadecimal is normally used.

An instruction to add 1 to the accumulator for the 8051 would look like the following in the ROM memory map 24 01 in hexadecimal Luckily, there are other methods. The instructions can be written in mnemonics, and software will assemble it directly into machine code. I1 Microcontroller ManualPart II V2.1 This code would be converted directly to 24 01. Writing code in mnemonics is easier but can still be very tedious. Assembler language, a low level programming language, is the next step up. While the majority of the code is still written in machine code mnemonics, there exist methods to use constants called symbols and the means to simplify coding. When programming the BS2 in PBASIC2, we used an application program Stampw.exe to edit our code. Stampw.exe then tokenized our code and transferred it to the BS2. From that point, with the exception of debugging data, the computer was no longer required. The PBASIC2 program on the PIC16C57 handled communications with the PC to accept the program being downloaded and to transfer it to EEPROM. Normally, a special piece of hardware is required to program microcontrollers such as the PIC16C57 and the Intel 8051. If you are familiar with EPROM programming devices, it is very similar. The microcontroller is set in a latchable socket ZIF socket, and the programming device handles accepting data from the computer and burning it into the microcontroller in much the same manner an EPROM or PROM is programmed. The binary machine code is transferred from the computer to the ROM space on the controller. Once programmed, the controller is placed in a circuit comprised of supporting interfacing electronics components. When power is applied, the microcontroller will begin reading ROM and executing the machine code instructions. With the BS2 we had the benefit of an activity board to allow us to write programs that communicated with input and output devices.

We could also have programmed the BS2 in the Activity Board, removed it and used it in a specialized circuit. Typically, any interfacing circuits will be of special design for the intended use of the controller. It would be difficult to provide students with everything needed to program a microcontroller directly and test it in a circuit for independent study. Luckily, there exist simulation programs for this. The one we will be using is UMPS from Virtual Micro Design. This package allows a PC to simulate a microcontroller for programming and operation. Programs such as UMPS are not just for training purposes. Professional programmers simulate programs to test and debug them. Programming in machine code can be cumbersome, and there are dozens of registers to keep track of. A good simulation program can give the programmer insights into what is happening internally with the controller and allow correct of bugs prior to burning the program into an expensive IC. While a license for a simulation package can cost hundreds to thousands of dollars, UMPS allows free distribution of a demonstration version with only limited reduced functionality. The associated labs outline these limitations. UMP also has a unique feature of allowing simulated devices resources to be connected to the microcontrollers. In Part II of the manual we will look at machine code, mnemonics, assembler and various fundamentals of microcontroller programming using the 8051. UMPS will be used to explore coding the 8051 in both mnemonics and assembler to read and control simulated input and output devices called resources. While it comes in different styles, such as the 8052, 8032 and so on, our discussion will focus on the 8051. This microcontroller has 4K bytes of internal ROM and 256 bytes internal RAM, 128 bytes of which are accessible for programming needs. Figure I1 shows the pinouts of the 8051. The majority of the pins are labeled PN.

b, where N is a port number a register byte and b is the bit number in the port byte. For example P1.5 this indicates bit 5 on port 1. These are comparable to P0P15 on theBS2. Many pins have

additional names in parenthesis. For example, P3.0 RXD is a special function pin that can be configured to capture incoming serial data in the RS232 format. The 8051 can also be configured to work with external RAM and ROM. The pins with labels of AD are used as address and data lines for external memory. Since there are a total of 16 address lines, how much external memory could the processor address. UMPS will be used to simulate operation of the 8051. Figure I2 is the Virtual Activity Board VAB designed for this manual using the UMPS component resources. Figure I3 is the schematic representation of the VAB. Note that all switches bring the inputs LOW to ground and that the switches are debounced by oneshots that are not shown. All the LEDs light on a HIGH. All LEDs have current limiting Resistors. The instructions are defined by a particular byte. The data that the instruction works with are bits, bytes or words. The instruction is known as the opcode. Each opcode has a unique byte that defines it to the microprocessor. In fact, there are many opcodes that have the same name, but have different associated bytes depending on the type of data they are working with. The data for the instruction is called the operand. This data may be a register, a bit in a register, a byte of data, or a 16bit word address. The second most important fact to understand is that of a register. We worked with registers on the BS2. The variables we declared were all held within these RAM registers. Microcontrollers have numerous registers many more than microprocessors. Lets take a look at a simple program. Etc D2 C2 SETB CLR A4 A3 I5 P2.4 P2.3 Microcontroller ManualPart II V2.1 Mnemonics are entered in the right side.

Upon moving to the next line, the code is assembled into machine language and updated on the left as bytes in hexadecimal. The memory location is the word of ROM in which that code is stored as bytes. In our program, the memory locations jump by 2s because most of the instructions took two bytes one for the opcode, and one for the operand. The only exception to this is the last line of LJMP 0000h, which used 3 bytes because the operand is a 16bit word using 2bytes. The Intel standard is to suffix the number with an h for hex or a b for binary. If the hex number starts with a letter AF, precede it with a zero. C5h a 0C5h. UMPS requires the 0 to be used, but will strip it out upon assembly. Figure I5 is the ROM memory map. Compare it to the CPU code window of I4. Can you see the same machine code in the locations specified. Lets analyze the program. Move the number 65h into the Accumulator. Move the value in the Accumulator into the P1 memory location. This will set pins P1.0 P1.7 equal to A 65h, our data display data lines. Make a long 16bit address jump to memory location 0000. The program will continue to loop. This means No Operation. It is used as fillers or for short delays 1 clock cycle. Table I1 Program I1 Breakdown Another window in the project displays certain registers and memory locations. Lets single step through the program and observe the contents of the registers affected by the program above. The program can be single stepped, one instruction at a time, by pressing F7 Trace Into. In Table I2 each change in a register is in bold. PC is the Program Counter. It is a 16bit word that points to the current address in memory of the instruction to be executed. On powerup or reset, the PC begins executing the instructions at memory location 0000h. What action did it have on the VAB. The LEDs corresponding to 1s lit and 0s stayed off. Also, the 7segment LEDs indicated the number 65. Not too bad, was it.

Note that we went through an accumulator to load a byte and transfer it. Almost all instructions operate by using registers to move and manipulate data. The most important register is the Accumulator Acc or A. It is the main arithmetic and logical register, and most math and logical operations are performed on data in the accumulator. If the first number is the instruction code, why are they different for the same MOV instruction. Its because they are very different instructions. Port 0 is now moved into the accumulator. It is now this value that is moved into P1. Referring to I3, P0 is connected to the eight input DIP switches. Clicking the GO button on the button bar allows the program to run. Changing the settings of the 8 DIP switches in the upperleft hand corner of the VAB changes the status of the 8 LEDs and the 7segment display. At some settings the 7segment display shows no numbers. This occurs when the associated nibble exceeds 9. The 4511 is a BDC to 7segment decoder. BCD is Binary Coded Decimal. Instead of being a binary number where the total

byte is a number, the high and low order nibbles define decimal numbers. For example 10010011 in binary would be 147 in decimal. In BCD, this digital value would equal 93. Its the same as converting it to hexadecimal, but neither nibble can exceed 9 or it would not be a valid decimal number. 7segments refer to the 7 sections of the display which are manipulated to show our numbers. I7 Microcontroller ManualPart II V2.1 RAM Memory Map and Registers. Figure I6 summarizes the 256 bytes of onchip RAM of the 8051. The lower 128 bytes 00h7Fh are registers and RAM available to the programmer. Locations 00h 1Fh are banks holding registers R0R7 in each. R0 R7 can be accessed faster than any other locations in RAM because their access is built into the instruction set for data manipulation in the same manner that accumulator was for MOV instructions. Only a single bank can be accessible at any one time.

If Bank 0 the default is specified, R0R7 will refer to RAM locations 00h07h. If Bank1 is specified, R0R7 will refer to memory locations 08h0Fh and so on. By having four banks, there are 36 actual bytes available for fast access. Different sections of a program may utilize different banks for the memory location. The active bank is selected as part of the Program Status Word register discussed later. This allows the programmer to perform logic operations on the specified bits and do other operations such as set SETB, clear CLR. This increases execution speed by not having to perform bit masking on bytes. The bits are numbered 00h 7Fh for a total of 128 individual bits. When memory is limited and operation must be maximized, the functionality of the resources such as these special registers are very important to programmers. I8 Microcontroller ManualPart II V2.1 Special Function Registers Special Function Registers SFRs start at address 80h. This section describes the purpose of the most important registers. Discussion of the other registers will follow as need arises. Accumulator Acc or A This is the main register in the controller. It is used for performing mathematical and logical operations. Program Status Word PSW In many instances where registers are concerned, the byte is not as important as the bits within it. Table I3 indicates the bits contained within the PSW, and Table I4 discusses their purpose. For the majority of the bits, certain instructions return results in the form of flags within the PSW. For example, with addition we may want to know if the operation resulted in exceeding the limits of our number system. The auxiliary carry flag AC in the PSW would indicate if this had occurred. CY AC FO RS1 RS0 OV P Table I3 PSW Bits Bit P OV RS0 RS1 F0 AC CY Discussion Even Parity flag, usually used in serial transmissions. If the total number of HIGH bits in a byte is EVEN, P will be set to 1, otherwise it will be 0. Not used Overflow flag.