1. Assembly Language Programming
2. Registers
3. Most registers in 8051 are 8 bits
4. Left is most significant bit, right is least significant
5. Data larger than 8 bits must be broken into 8 bit chunks
6. Most common registers
   1. Accumulator (A)
   2. B
   3. R0-7
   4. DPTR (data pointer)
   5. PC (Program counter)
   6. DPTR and PC are only non-8 bit registers
7. Accumulator used for all arithmetic and logic instructions
8. MOV
   1. Copy data from one location to another
   2. MOV destination, source
   3. Destination and source have same value after operation
   4. # signifies value instead of address
   5. You need a 0 before a letter when specifying values, ie “#0F9H”
   6. Error if moved value is greater than 8 bits
   7. Value w/o pound sign means load from memory location
9. ADD
   1. ADD A, source
   2. Add source byte to A and put result in A
   3. Destination must always be accumulator
10. Assembly language Format
    1. Uses mnemonics for machine codes
    2. Low level, deals directly with structure of CPU
    3. Assembler translates code into machine code
    4. Lines contain a mnemonic, one or two operands
    5. Optional directives give directions to assembler
    6. ORG tells assembler where to place opcode in memory
    7. END indicates assembler to end source code
    8. Mnemonics and operands generate machine code
    9. Directives do not generate machine code, they only are used by assembler
11. Assembling and Running an 8051 Program
    1. Code written in an editor
    2. Save with asm or src
    3. Asm file is fed to assembler which converts instructions to machine code and produces object file and list file
    4. Assemblers link program takes one or more object files and produces absolute object file with extension abs
    5. Abs file fed into the OH (object to hex) which creates a hex file that can be burned into ROM
    6. List file lists all opcodes and addresses as well as errors assembler detected.
12. Program Counter
    1. Points to address of next instruction to be executed
    2. When CPU fetches opcode from program ROM, program counter incremented to point to next instruction
    3. PC is 16 bits wide
    4. For 8051, PC wakes up to memory address 0000 when powered up
    5. Power up is applying Vcc to reset pin
    6. Program is stored in RAM addresses alternating between opcode and operand
    7. Once instruction is finished, program counter is incremented to next instruction
    8. Some microcontrollers have as little as 4K bytes ROM, none in 8051 family have any more than 64K.
13. Data Types and Directives
    1. 8051 has only one data type that is 8 bits
    2. DB: define byte, used to define 8 bit data
    3. B required after number for binary, H for hexadecimal
    4. Use quotations to define as ASCII string
    5. EQU defines a constant without occupying a memory location
    6. By using EQU, if a constant used throughout a program needs to be changed, the programmer only needs to change it once
    7. END indicates the last line of an asm file
    8. Labels must be unique, have letters, numbers, ?, ., @, \_ , $ and must start with a letter
    9. They can’t use reserved words
14. Flag Bits and PSW
    1. PSW: Program status word
       1. Flag register for arithmetic conditions
       2. Only 6 bits are used, last two are user definable
       3. Four flags are conditional: CY (carry), AC(auxiliary carry), P(parity), OV(overflow)
       4. PSW.3 and PSW.4 for changing bank registers
       5. PSW.5 and PSW.1 are general purpose
       6. Carry: When carry out of D7 it (addition or subtraction). Can also be directly set or cleared
       7. Auxiliary Carry: Carry from D3 to D4 during ADD or SUB.
       8. Parity: Number of 1’s in A. Odd number of 1’s, P=1. Even number of 1’s, P=0
       9. Overflow: Set when result of signed number operation is too large, causing high order bit overflow into sign bit
15. Register Banks and Stack
    1. 8051 has 128 bytes of RAM
    2. Assigned addresses 00 to 7FH
    3. 00 to 1F are for register banks and stack
    4. 20H to 2FH for bit addressable read/write memory
    5. 30H to 7FH are read and write storage (scratch pad)
    6. 32 bits of RAM for registers, 4 banks of 8 registers
    7. Bank 1 of registers uses same RAM space of stack
    8. Register bank 0 is default
    9. Bits D4 and D3 of PSW used to select register bank
16. Stack
    1. Used to store information temporarily
    2. Stack pointer is 8 bits wide
    3. By default contains value 07
    4. 08 is first value used by stack
    5. Storing in stack is PUSH, pulling contents off is POP
    6. Stack pointer points to last used location of stack and is incremented by one as more data is pushed onto stack
    7. RAM addresses must be used to push onto stack
    8. When there is a POP, top byte of stack copied to register specified and stack pointer decremented
    9. If we need more than 24 bytes of stack, SP can be changed to point to RAM locations 30 – 7FH by doing a MOV to SP, #XX
    10. Stack also saves address of instruction just below CALL instruction to know where to resume from subroutine
    11. If a program is using register banks 1 and 2, we can reallocate another section of RAM to stack.
17. JUMP, LOOP, and CALL
18. Looping

1. DJNZ reg, label

2. Register is decremented and jump back to target address occurs

3. Value for count loaded into register specified

4. Because registers are 8 bits, highest possible loop value is 256 times.

5. To loop more than that, nested loops must be used.

B. Conditional Jumps

1. JZ: Jump if A = 0.

2. JNZ: Jump if A =/=0

3. JNC: Jump if CY = 0

4. JC: Jump if CY = 1

5. All conditional jumps are short jumps, address must be within +-128 bytes of program counter

C. Unconditional Jumps

1. LJMP: long jump.

2. 3 byte instruction that allows for 16 bit address of target location.

3. Only used when necessary because there is not much ROM

4. SJMP is 2 byte instruction to save bytes

5. Short jumps can be 127 bytes forward or 128 bytes backward

D. Call

1. Used to call subroutine, a task performed frequently

2. LCALL: 3 bytes, 16 bit target address

3. Processor automatically saves address of instruction after LCALL on stack.

4. RET transfers control back to caller, this must be last instruction of subroutine.

5. There must be equal number of PUSH and POP functions in a subroutine to avoid losing track of address of next instruction

6. For LCALL, address can be anywhere in 64K memory.

7. ACALL is a 2 byte instruction, address must be within 2K bytes of current address.

8. No difference in terms of use of stack

E. Time Delay

1. CPU takes certain number of clock cycles to execute instruction

2. Delay subroutine requires setting a counter and using a loop

3. Using NOP instruction in loop increases length of delay

4. Crystal frequency and design of microcontroller are factors in length of delay.

III. I/O Port Programming

1. I/O Port Programming

1. 4 ports, 32 pins for these ports

2. Configured as inputs by default

3. When first 0 written to port, it becomes output. Must be reconfigured by sending a 1.

4. In order to use Port 0 as both input and output ports, must be connected to 10K pull up resistor

5. Ports 1,2, and 3 don’t require pull up resistors

6. All ports have FFH on them at reset

B. I/O Bit Manipulation

1. 8051 allows individual bits of ports to be accessed

2. Use “SETB X.Y”

3. SETB used for setting bit to 1, CLR to make bit 0, CPL to change to opposite of current

4. JNB and JB used to monitor input bits

5. Carry flag can be used to save bit values

6. In reading ports, you can either read status of input pin or read internal latch of output port

7. Write 1 to a bit to make it an input, only certain instructions can access external data present at pins.

IV. Addressing Modes

1. Immediate

1. Source operand is constant

2. Immediate data must have pound sign

3. Can be used to load info into registers including DPTR

4. DPTR can be accessed as two 8 bit registers: DPH (high byte), and DPL ( low byte)

B. Register addressing mode

1. Use of registers to hold data to be manipulated

2. Size of source and destination have to match

3. Movement of data between Rn registers not allowed

C. Direct addressing mode

1. Most often used to access RAM locations 30 – 7FH

2. Known RAM address is used, given as part of instruction.

3. A, B, PSW, and DPTR are Special function registers with their own

addresses.

4. Only direct addressing used for pushing onto stack, or popping off stack.

D. Register Indirect Addressing Mode

1. Register used as pointer to data

2. For data in CPU, only R0 and R1 can be used

3. Must be preceded by @ sign

4. Allows access of data to be dynamic rather than static

5. Allows for access of data through looping

6. To access data in external RAM or on –chip ROM, 16 bit pointer (DPTR) is needed

7. Often used for look up tables in ROM

8. Instruction is “MOVC A, @A+DPTR”

9. DPTR and A both form address for element in on chip ROM

10. Because data is in program space ROM, MOVC is used instead of MOV.

11. 8051 has extra 64K set aside as data storage. This is called external memory, accessed by MOVX instruction.

12. Data space cannot be shared between code and data.

E. Bit Addressable RAM

1. 16 bytes of RAM are bit addressable (20H to 2FH)

2. Can be referred to by byte addresses 00 – 7FH

F. Bit addressable Ports

1. Any bit of the four ports can be accessed using SETB

2. P0-P3 registers are bit addressable

G. Bit addressable Registers: Only A, B, PSW, IP, IE, ACC, SCON, and TCON are bit addressable