EE 354 **August 31, 2011**
Notes on Addressing modes

An addressing mode is the method whereby the operand of an instruction specifies the data. There are 7 generic addressing modes and many more addressing mode combinations.

Generic Addressing modes

Direct Addressing – in direct addressing the operand is the address in memory where the data is located.

Immediate Addressing – for immediate addressing the operand is the data.

Register Addressing – for register addressing the operand is the address of a register that has the data.

Register Indirect Addressing – An indirect address is the address of the address of the data. In register indirect addressing, the operand is the address of a register that holds the memory address of the data.

Memory Indirect Addressing – for this mode the operand is the address of a memory location that holds the address of the data. This mode is not commonly used.

Indexed Addressing – in indexed addressing the operand is added to an index register and the sum forms the address in memory of the data.

Relative Addressing – this mode is used mostly for branching instructions. The operand is added to the program counter to form the address of the next instruction.

Variations

There are numerous variations on addressing modes and some machines list as many as 20 to 30. For example, indexed addressing with auto-increment is a variation. Here the operand is added to an index register and the result is the address in memory of the data. Once this is complete the index register is automatically incremented. This is also available on some machines with auto-decrement. Another variation is *base register addressing*. This is identical to indexed addressing except the index register is called a base register. For indexed addressing we normally increment or decrement the index register but in base register addressing we increment or decrement the operand so the base register remains constant. Also, some machines implement this mode so they can do *base register indexed addressing* in which the operand is added to an index register plus a base register with the sum of the three forming the address of the data in memory. The Intel 8086 implements this mode. Other variation involve different data types such as strings, Boolean, or bits.

8051 Addressing modes

The 8051 and its variants implement *direct, immediate, register, register indirect, indexed, and relative* addressing modes. Because some of these addressing modes are limited to certain registers (such as the accumulator or R1 and R0) or they are limited to certain forms of memory (such as code, data, or external) the actual 8051 addressing modes have a number of variations. The 8051 addressing modes are complicated by the fact that registers are really part of the data memory address space. The instruction `mov R0, 1` and `mov 0, 1` do exactly the same thing if you are using register bank 0 since memory location 0 is the same as register 0.

It is difficult to make generalizations about the addressing modes on the 8051 because all addressing modes do not apply to all instructions. For example, you can move one data memory location into another with an instruction like mov 0, 1 but you cannot add to memory locations. The instruction add 0, 1 is illegal.

For a list of instructions and their address mode variations. See the text on pp.247-294 for a detailed description of every instruction and its addressing mode variations. In addition pp. 244-246 summarizes the instructions by op code.

The list below presents some generalizations that apply to most instructions.

1. For the arithmetic and logical group of instructions one operand must be the A register. The second operand can be any register R0 to R7, an indirect register R0 or R1, or 8-bit immediate data.
Examples:

```
add A, #55h;  55h is the data
add A, 22;    memory location 22
add A, R4
add A, @R1;   R1 has the memory address of the data
```

2. Increment and decrement can operate on any register R0-R7 plus A and can be used with the indirect R0 and R1 registers. Direct addressing of data memory also works.
Examples:

```
inc A
inc R4
inc @R0;  Can only use R0 and R1 for indirect
inc 22;   increments memory location 22
```

3. You can move immediate data into the any register or A or the indirect registers R0 and R1. You can also move memory to memory direct for the first 256 bytes (8-bit address). You can move 16-bit immediate data into the data pointer.
Examples:

```
mov A, #4
mov R3, #34
mov @R0, #56
mov dptr, #data16
```

4. You can move any 8-bit memory location to any register or A or vice-versa. This also works with R1 and R0 in indirect mode. you cannot do a register to register move but you can move memory to memory for the first 256 bytes of memory.
Examples:

```
mov A, 54;
mov R4, 32
mov 3, 4;    memory to memory
mov 54, R3
mov @R0, 56; memory location 56 to location in R0
```

5. There are two absolute jumps to address locations. All of the others are relative jumps. Examples:

```
ljmp 1234h;  long jump absolute
ajmp target; absolute jump to an 11 bit address.
sjmp target; relative jump -128 to +127 of present location
jnz target;  jump not zero relative
jc target;   jump no carry relative
```

6. Call instructions can be conditional but otherwise work like jumps.