LECTURE 18

Program Branching Instruction Set

Topics to be covered

- Program Branching Instruction

Program Branching Instructions

- Program branching instructions are used to control the flow of actions in a program
- Some instructions provide decision making capabilities and transfer control to other parts of the program, e.g. conditional and unconditional branches

Mnemonic	Description
ACALL addr11	Absolute subroutine call
LCALL addr16	Long subroutine call
RET	Return from subroutine
RETI	Return from interrupt
AJMP addr11	Absolute jump
LJMP addr16	Long jump
SJMP rel	Short jump
JMP @A+DPTR	Jump indirect
JZ rel	Jump if A=0
JNZ rel	Jump if A NOT=0
CJNE A,direct,rel	Compare and Jump if Not Equal
CJNE A,#data,rel	
CJNE Rn,#data,rel	
CJNE @Ri,#data,rel	
DJNZ Rn,rel	Decrement and Jump if Not Zero
DJNZ direct,rel	
NOP	No Operation

ACALL addr11

- This instruction **unconditionally** calls a subroutine indicated by the address
- The operation will cause the PC to increase by 2, then it pushes the 16-bit PC value onto the stack (low order byte first) and increments the stack pointer twice
- The PC is now loaded with the value *addr11* and the program execution continues from this new location
- The subroutine called must therefore start within the same 2 kB block of the program memory

- No flags are affected

- *Example*:

```
ACALL   LOC_SUB
```

LCALL addr16

- This instruction calls a subroutine located at the indicated address
- The operation will cause the PC to increase by 3, then it pushes the 16-bit PC value onto the stack (low order byte first) and increments the stack pointer twice
- The PC is then loaded with the value *addr16* and the program execution continues from this new location
- Since it is a Long call, the subroutine may therefore begin anywhere in the full 64 kB program memory address space
- No flags are affected
- *Example*:

```
LCALL   LOC_SUB
```

- This instruction returns the program from a subroutine
- RET pops the high byte and low byte address of PC from the stack and decrements the SP by 2
- The execution of the instruction will result in the program to resume from the location just after the “call” instruction
- No flags are affected
- Suppose SP=0BH originally and internal RAM locations 0AH and 0BH contain the values 30H and 02H respectively. The instruction leaves SP=09H and program execution will continue at location 0230H

RETI

- This instruction returns the program from an interrupt subroutine
- RETI pops the high byte and low byte address of PC from the stack and restores the interrupt logic to accept additional interrupts
- SP decrements by 2 and no other registers are affected. However the PSW is not automatically restored to its pre-interrupt status
- After the RETI, program execution will resume immediately after the point at which the interrupt is detected
- Suppose SP=0BH originally and an interrupt is detected during the instruction ending at location 0213H