Chapter 4

8051 Family Microcontrollers Instruction Set

Lesson 5

Program Flow Control and Interrupt Flow Control Instructions

Branch instructions- Jump to new value of Program Counter (PC)

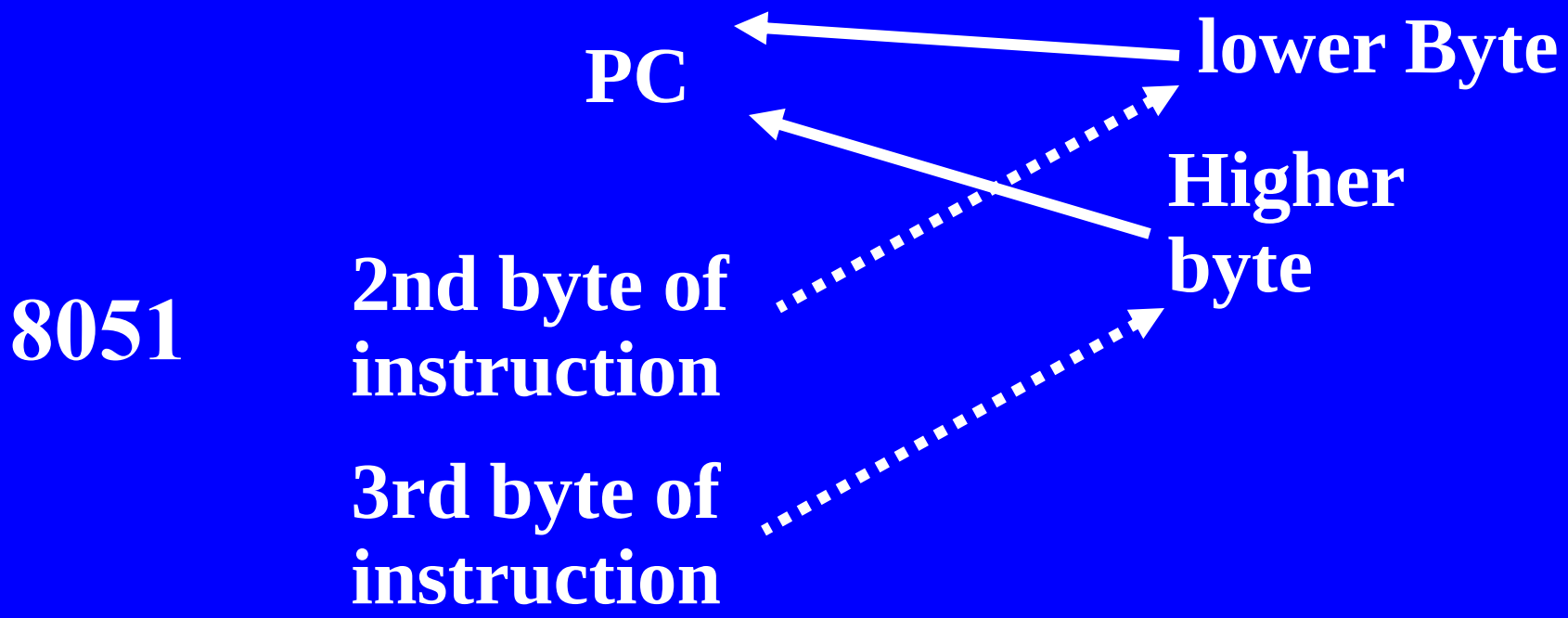
LJMP address16

AJMP addr11

SJMP rel

JMP A, @A + DPTR

LJMP Addr16


PC
lower Byte
Higher byte
2nd byte of instruction
3rd byte of instruction
8051

LJMP Instruction Execution

2 clock cycles

STEP 1 — Fetch opcode-bits

STEP 2 — Fetch lower byte for branch address

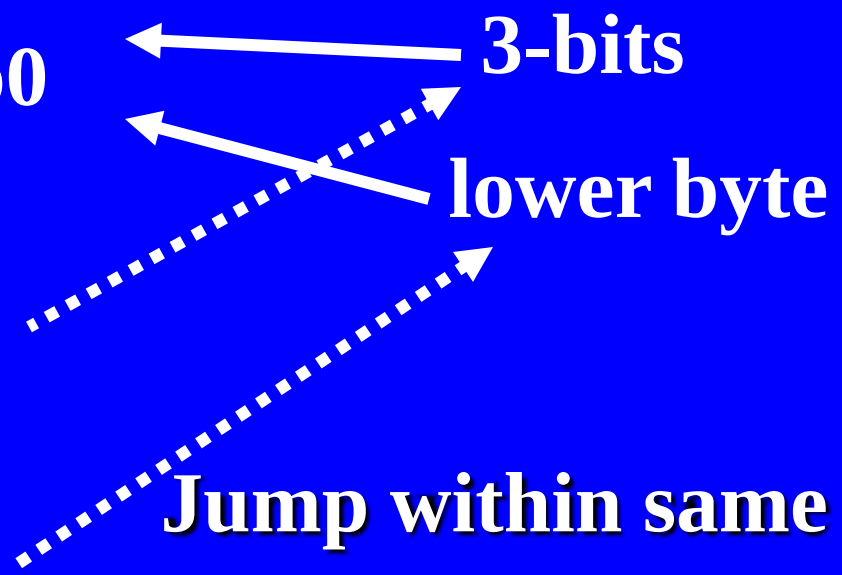
STEP 3 — Fetch higher byte for branch address

PCL

PCH

Time

AJMP Addr11


PC bit10-b0
3-bits
lower byte
8051
3-bits from1st byte
2nd byte of instruction
Jump within same segment of 2 kB.
Upper 5-bits of PC unchanged

AJMP Instruction Execution

2 clock cycles

STEP 1 — **Fetch five opcode-bits**

STEP 2 — **Fetch branch address 3 bits b10-b9-b8**

STEP 3 — **Fetch lower byte b7--... - b0 for branch address**

PCH 3-bits

PCL

Time

SJMP rel

8051

Jump relative after adding a two's complement number which is within −128 to +127 of next instruction

2nd byte of instruction a number in 2's complement form ⇢ Add **in Next instruction's PC** ↓ **PC bit15-b0**

SJMP Instruction Execution

2 clock cycles

STEP 1 **Fetch opcode-bits**

STEP 2 **Fetch 8 bits b7--b0 for +or -ve number**

Time

PCH

PCL

Add ..... PCH PCL

Next instruction

Branch to new address

Example- SJMP F8H

Rel = 11111000 mean number = −8

Let PC before the SJMP = 1F0CH

Next instruction PC when no jump = 1F0EH

PC ← PC − 8.

Therefore, when jump

the new PC ← 1F06H

JMP @ A + DPTR

PC bit15-b0

Pointed instruction PC

8051

8-bits from *A*

16-bit from *DPTR*

Add; DPTR is pointer to the code block and A is offset

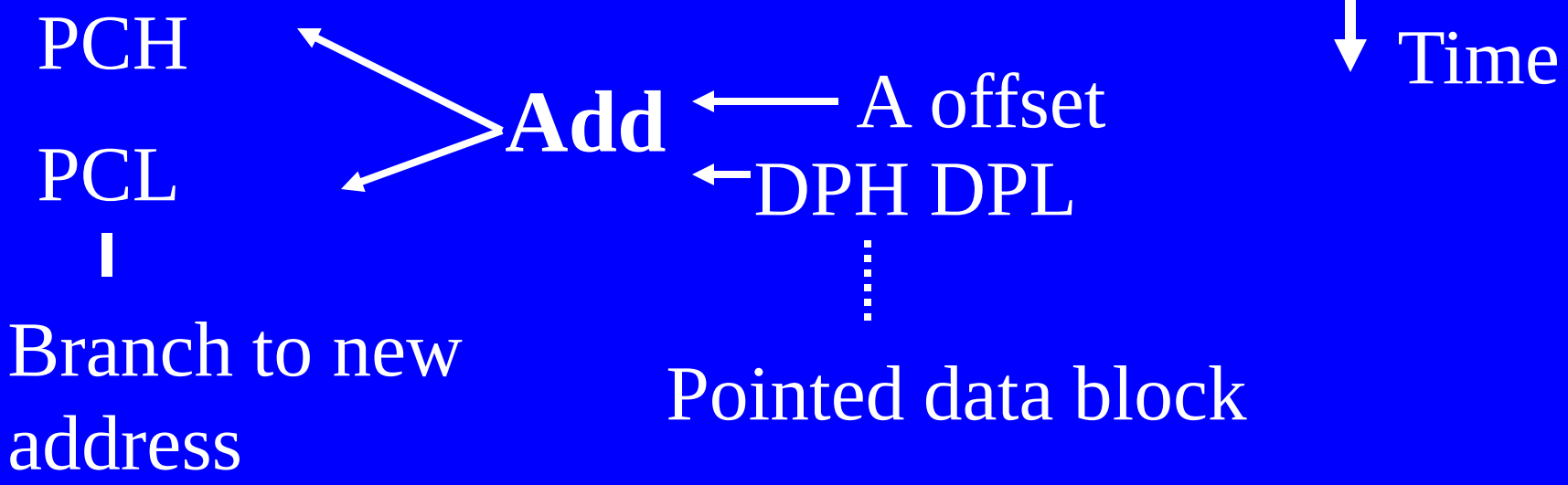
Add offset at A in DPTR to find code address, jump to the memory address

JMP @ A + DPTR JMP Instruction Execution

2 clock cycles

STEP 1

Fetch 8 opcode-bits



Example- JMP@ A+ DPTR

DPTR = 11ADH; A = 08H

Let PC after the JMP = 1F0CH

This PC replaces and gets new 16-bits

PC ← A + DPTR.

Therefore, new PC ← 11B5H

Conditional jumps

JNZ rel;
JZ rel;
JC rel;
JNC rel;

JZ rel; jump on zero *A*

8051

If A= 00H No

PC bit15-b0 ← Next instruction PC

If A= 00H Yes

2nd byte of instruction → Add in PC the 2's complement number

↓

Jump relative after adding a two's complement number within −128 to +127 of next instruction

JNZ rel; jump on not zero *A*

8051

If A= 00H Yes

PC bit15-b0 ← Next instruction PC

If A= 00H No

2nd byte of instruction → Add in PC the 2's complement number

↓

Jump relative after adding a two's complement number within −128 to +127 of next instruction

8051

JC rel; jump on C (PSW.7) not zero

If C= 0 Yes

PC bit15-b0 ← Next instruction PC

If C= 1 Yes

2nd byte of instruction → Add in PC the 2's complement number

↓

Jump relative after adding a two's complement number within −128 to +127 of next instruction

8051

JNC rel; jump on C (PSW.7) not 1

If C= 1 Yes

PC bit15-b0 ← Next instruction PC

If C= 0 Yes

2nd byte of instruction → Add in PC the 2's complement number

↓

Jump relative after adding a two's complement number within −128 to +127 of next instruction

Example- JZ F8H

Rel = 11111000 mean number = − 8

Let PC before the JMP = 1F0CH

Next instruction PC when no jump = 1F0EH

PC ← PC − 8.

Therefore, new PC 1F06H if A = 00H,else 1F0EH

JB bit, rel
JNB bit, rel
JBC bit, rel;

JB *bit*, rel; jump on bit at address of bit is set 1;

If bit at address of bit = 0 Yes

PC bit15-b0 ← Next instruction PC

If bit at address of bit = 1 Yes

8051 2nd byte of instruction → Add in PC the 2's complement number

↓

Jump relative after adding a two's complement number within −128 to +127 of next instruction

JNB *bit*, rel; jump on bit at address of bit is reset 0;

If bit at address of bit = 1 Yes

PC bit15-b0 ← Next instruction PC

If bit at address of bit = 0 Yes

8051 2nd byte of instruction → Add in PC the 2's complement number

↓

Jump relative after adding a two's complement number within −128 to +127 of next instruction

JBC *bit*, rel; jump on bit is set 1 and then reset C = 0

If bit at address of bit = 0 Yes

PC bit15-b0 ← Next instruction PC

If bit at address of bit = 1 No

2nd byte of instruction → Add in PC the 2's complement number and reset C

8051

Jump relative after adding a two's complement number within −128 to +127 of next instruction and reset C = 0

Example— JB 91H, F8H

bit = 91H mean bit1 Port P1

Rel = 11111000 mean number = -8

Let PC before the JMP = 1F0CH

Next instruction PC when no jump = 1F0EH

PC ← PC −8.

Therefore, new PC 1F06H if P1.1 = 1,else 1F0EH

Subroutine Call Instructions

Save PC on stack and then call the instruction of the routine at new value of Program Counter (PC)

LCALL address16

ACALL addr11

Execution of Subroutine Call Instructions

Saving PC on stack before Branch instruction-call routine at new value of Program Counter (PC)

Before call - (1) Increment SP
and then move the next instruction PCL byte to SP pointed address (2) Increment SP and then move the next instruction PCH byte to SP pointed address

LCALL Addr16

2nd byte of instruction lower Byte

PC

Higher byte

8051

3rd byte of instruction

Next instruction PC saves on stack at address pointed by SP + 1 and SP + 2.

New SP becomes SP+02H

LCALL Addr16 Instruction Execution

2 clock cycles

STEP 1 **Fetch opcode-bits**

STEP 2 **Fetch lower byte for call address**

STEP 3 **Fetch higher byte for call address**

Step 4: Old PCL after step 3 at SP +1

Step 5: Old PCH after step 4 at SP +2

Step 6:New PCL

Step 7: New PCH

Time

Return from call instruction - RET

Let SP = 07H Let PC before the CALL = 1F0CH.

PC after the ACALL = 1F0FH.

Call instruction stacks 1F0EH at 08-09H.

08H ← 0F and 09H ← 1F

Return retrieves PC. PCH from 09H, PCL from 08H. PC again = 1F0FH

ACALL Addr11

Call within same segment of 2 kB. Upper 5-bits of PC unchanged

Next instruction PC saved on stack. SP is now SP+02H

ACALL Addr11 Instruction Execution

2 clock cycles

STEP 1 **Fetch opcode-bits and 3 bits for PC**

STEP 2 **Fetch lower byte for call address**

Time

Step 6: Replace 3 bits to get New PCH

Step 7: Replace all 8 bits to get new PCL

Step 4: Old PCL after step 2 at SP +1

Step 5: Old PCH after step 3 at SP +2

Example— ACALL codes are 91H-F0H

3 higher bits in first byte = 100 and 8-bits in 2nd byte of instruction = 11111000.

Let SP = 07H

Let PC before the CALL = 1F0CH

Next instruction PC= 1F0EH. It stacks 1F0CH at 08-09H. Five bits are 00011

PC ← PC + 00011 1001111 0000. Therefore, new PC ← 3BFEH

Return from call instruction - RET

Let SP = 07H Let PC before the CALL = 1F0CH.

PC after the ACALL = 1F0EH.

Call instruction stacks 1F0EH at 08-09H.

08H ← 0E and 09H ← 1F

Return retrieves PC. PCH from 09H, PCL from 08H. PC again = 1F0EH

Decrement and Jump conditional Combined instructions

DJNZ Rn, rel;
DJNZ direct, rel

DJNZ Rn, rel; jump on byte at Rn after decrement = not 0;

Decrease Rn, If byte at Rn after decrement = 0

Yes PC bit15-b0 ← Next instruction PC

If byte at Rn after decrement not 0 Yes

8051 2nd byte of instruction → Add in PC the 2's complement number

↓

Jump relative after adding a two's complement number within −128 to +127 of next instruction

DJNZ direct, rel; jump on byte at direct address after decrement = not 0;

Decrease byte at direct address, If byte becomes 0

Yes PC bit15-b0 ← Next instruction PC

If byte remains not 0

Yes 2nd byte of instruction → Add in PC the 2's complement number

8051

↓

Jump relative after adding a two's complement number within −128 to +127 of next instruction

Compare and then conditional jump

CJNE A, #data, rel;
CJNE Rn, #data, rel;
CJNE @Ri, #data, rel;
CJNE A, direct, rel;

CJNE A, #data, rel; jump on byte at A on comparison not equal

Compare A and data byte, if equal

Yes PC bit15-b0 ← Next instruction PC

Compare A and data byte, if not equal Yes

8051 2nd byte of instruction → Add in PC the 2's complement number

↓

Jump relative after adding a two's complement number within −128 to +127 of next instruction

CJNE Rn, #data, rel; jump on byte at Rn on comparison not equal

Compare Rn and data byte, if equal

Yes PC bit15-b0 ← Next instruction PC

Compare Rn and data byte, if not equal Yes

8051 2nd byte of instruction → Add in PC the 2's complement number

↓

Jump relative after adding a two's complement number within −128 to +127 of next instruction

CJNE @Ri, #data, rel; jump on byte pointed by Ri on comparison not equal

Compare byte pointed by Ri and data byte, if equal Yes PC bit15-b0 ← Next instruction PC

Compare byte pointed by Ri and data byte, if not equal Yes

2nd byte of instruction →

Add in PC the 2's complement number

8051

↓

Jump relative after adding a two's complement number within −128 to +127 of next instruction

CJNE A, direct, rel; jump on byte at direct on comparison not equal to A

Compare byte at direct and byte at A, if equal

Yes PC bit15-b0 ← Next instruction PC

Compare byte at direct and byte at A, if not equal Yes 2nd byte of instruction → Add in PC the 2's complement number

8051

↓

Jump relative after adding a two's complement number within −128 to +127 of next instruction

Delay and Interrupt flow control— NOP and RETI

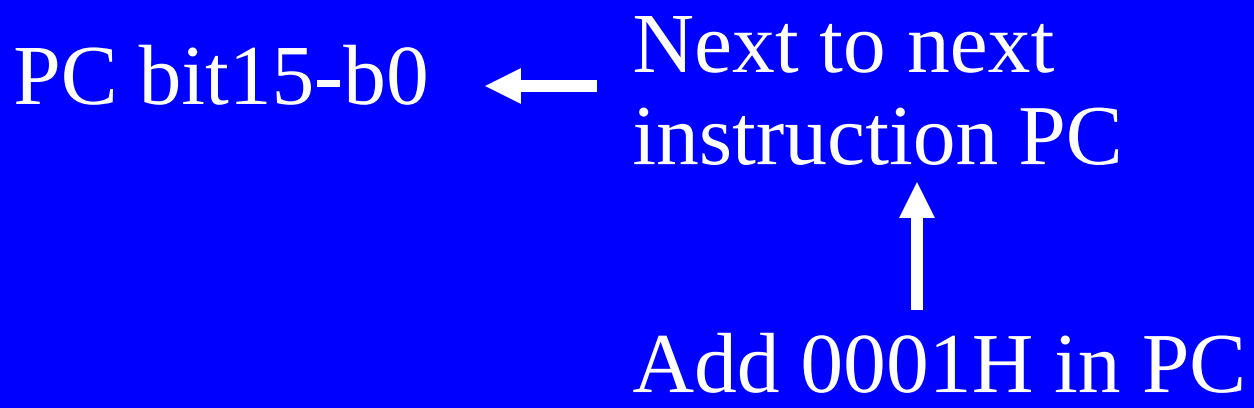
NOP instruction

Let PC before the NOP = 1F0CH

After NOP, PC ⟵ PC +1 1F0DH

PC ⟵ 1F0DH

NOP



8051 NOP instruction executes delay = 1 cycle = 1 μs for 12 MHZ XTAL

8051

100 times executed in a loop NOP instruction delay = 100 cycle =100 μs for 12MHZ XTAL

NOP Instruction Execution

STEP 1

Fetch opcode-bits = 00000000

1 clock cycle

Time

Branch is to next to next address after NOP

Return from interrupt ISR— RETI

Let SP = 77H

Let PC before the ISR CALL = 1F0CH

ISR call stacks 1F0EH at 08-09H. 78H ← 0E and 79H ← 1F

Return retrieves PC of last interrupted ISR.

Return retrieves PC if no interrupt arose during ISR running then the PCH from 79H, PCL from 78H. PC again = 1F0EH. SP = 77H.

Summary

We learnt 8051 family program flow control instructions

- AJMP,LJMO, SJMP rel,
- conditional jumps
- decrement and then test and jump
- increment and then test and jump
- LCALL,ACALL, RET
- NOP and RETI