Application Notes

1 Instruction

EM78xxx is a RISC-like single chip microcontroller. Each instruction is a 13-bit word divided into an OP code and one or more operands. All instructions are mostly executed with single instruction cycle, unless the program counter is changed by executing the "MOV R2, A", "ADD R2, A", "LCALL", or "LJMP" instruction. In this case, the execution takes two instruction cycles.

In addition, the EM78xxx has the following characteristics:

- Each bit of any register can be set, cleared, or tested directly.
- Any I/O Register can be accessed as a general-purpose register. That is, the same instruction set that accesses the general-purpose register can operate under I/O register.

2 Quick Reference to Instruction Details

(Continuation)

Instruction	Description	Go to Page
IOW	Move the A Register to IOCR	4
JBC	Bit Test, Skip if Clear	19
JBS	Bit Test, Skip if Set	19
JMP	Unconditional Branch	20
JZ	Increment R, Skip if “0”	18
JZA	Increment R, Place in the A Register, Skip if “0”	18
LCALL	Subroutine Call	22
LJMP	Unconditional Branch	23
MOV	Move Data	7
NOP	No Operation	2
OR	Inclusive OR	10
PAGE	Switch Page of ROM	22
RET	Return from Subroutine	5
RETI	Return from Interrupt	5
RETL	Return Immediate Data to A Register	6
RLC	Rotate Left R through Carry	17
RLCA	Rotate Left R through Carry, Place in the A Register	17
RRC	Rotate Right R through Carry	16
RRCA	Rotate Right R through Carry, Place in the A Register	16
SLEP	Seep	3
SUB	Subtract	8
SWAP	Swap R	18
SWAPA	Swap R, Place in the A Register	17
TBRD	Table Read	24
WDTC	Clear Watchdog Timer	4
XOR	Exclusive OR	11

3 Instruction Description

■ NOP (No Operation)

Syntax	NOP
Operation	No Operation
Status Affected	None
Description	No operation. NOP is used for time delay

Example	P50 output a 3μs pulse (system clock = 2MHz):
	`BS 0x5,0x0 ;P50 output high`
	`NOP ;Delay 2 instruction cycles`
	`NOP`
	`BC 0x5,0x0 ;P50 output low`

■ DAA (Decimal Adjust)

Syntax	DAA
Operation	if [A<3 : 0> > 9].OR.[DC=1] then A<3 : 0> + 6 → A<3 : 0>; if [A<7 : 4> > 9].OR.[C=1] then A<7 : 4> + 6 → A<7 : 4>;
Status Affected	C
Description	DAA adjusts the eight-bit value in the accumulator resulting from an earlier addition of two variables (each in packed-BCD format), and produces two four-bit digits.

Example	`Perform a decimal 6+9 operation:`
	`MOV A,@0x6`
	`MOV 0x10,A`
	`MOV A,@0x9`
	`ADD A,0x10 ;A = 0xf`
	`DAA ;A = 15H (packed BCD)`

■ CONTW (Move A to CONT)

Syntax	CONTW
Operation	A → CONT
Status Affected	None
Description	Move data from A register to CONT register

Example	`MOV A,@0x83`
	`CONTW ;0x83 = 10000011`
	`;Prescaler is 1:16`
	`;Assign prescaler to TCC`
	`;TCC increment from low to high`
	`;Transition on TCC pin`
	`;Internal clock for TCC signal`
	`;Disable interrupt`
	`;Port 7 internal pull-high disable`

■ SLEP (Sleep)

Syntax	SLEP
Operation	00h → WDT 0 → WDT Prescaler 0 → P 1 → T
Status Affected	P, T
Description	Watchdog Timer is reset. If prescaler is assigned to the WDT, the prescaler is reset. The Power-down Status Affected (P) is cleared and the Time-out Status Affected (T) is set. The processor goes into SLEEP mode with the oscillator stopped. See data sheet on SLEEP mode for more details.

Example	`SLEP ;The processor is put into SLEEP mode`

■ WDTC (Clear Watchdog Timer)

Syntax	WDTC
Operation	00h → WDT 0 → WDT Prescaler 1 → P 1 → T
Status Affected	P, T
Description	Watchdog Timer is reset. If the prescaler is assigned to the WDT, the prescaler is reset. The Power-down Status Affected (P) is set. The Time-out Status Affected (T) is set.

Example	`WDTC ;Clear the Watchdog Timer counter`

■ IOW (Move the A Register to IOCR)

Syntax	IOW R
Operation	A → IOCR
Status Affected	None
Description	Move data from A register to Control Register IOCR

Example	Set the odd bits of Port 6 to internal pull-high mode. The control register of Port 6 is I/O port control register 0xd:
	`MOV A,@0xaa`
	`IOW 0xd`

■ ENI (Enable Interrupt)

Syntax	ENI
Operation	1 → INT
Status Affected	None
Description	Enable interrupt by setting the INT bit. The INT is a global interrupt enable bit. There are several interrupt sources, e.g., internal TCC overflow interrupt, external INT pin interrupt, port pin changed interrupt, and so on. When one of these interrupts occurs, the processor will perform an interrupt phase. First, it will push the present PC into the top of stack and reset INT flag to disable further interrupt. Then it will set the PC to corresponding interrupt vector, fetch the related code, and execute. The Interrupt Status Register (register 0xf) indicates the interrupt source.

Example	`ENI ;Enable interrupt`
	`After Instruction`
	`Bit 6 of CONT register : 1`

■ DISI (Disable Interrupt)

Syntax	DISI
Operation	0 → INT
Status Affected	None
Description	Disable interrupt by clearing the INT bit. The INT is a global interrupt enable bit.

Example	`DISI ;Disable interrupt`
	`After Instruction`
	`Bit 6 of CONT register : 0`

■ RET (Return from Subroutine)

Syntax	RET
Operation	[Top of Stack] → PC
Status Affected	None
Description	Return from subroutine. Stack is popped and the top of the stack is loaded into the PC.

Example	`TEST:`
	`  •`
	`  •`
	`  •`
	`  RET`
	`  •`
	`  •`
	`  •`
	`  CALL TEST`
	`HERE ADD  A,@0x1`
	`  •`
	`  •`
	`  •`
	`Before Instruction`
	`  Top of Stack = address HERE`
	`After Instruction`
	`  PC = address HERE`

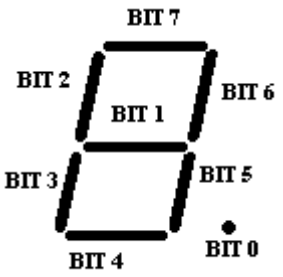
■ RETI (Return from Interrupt)

Syntax	RETI
Operation	[Top of Stack] → PC 1 → INT
Status Affected	None
Description	Return from interrupt routine. Stack is popped and the top of the stack is loaded into the PC. The interrupt is enabled by setting the INT bit. The INT is global interrupt enable bit.

Example	INTRTN: ;Interrupt routine • • • RETI MAIN: • • • ;Interrupt occurs HERE: • • • Before Instruction [Top of Stack] = address HERE After Instruction PC = address HERE INT flag = 1

■ RETL (Return Immediate Data to A Register)

Syntax	RETL k
Operation	k → A [Top of Stack] → PC
Status Affected	None
Description	Return from subroutine. The immediate data "k" is loaded into the A register. Stack is popped and the top of the stack is loaded into the PC

Example	Perform a 7-segment LED translation table. The 7-segment LED connected to Port 6. ;Define register PC == 2	
	TRANS: ADD PC,A RETL @0b11111100 RETL @0b01100000 RETL @0b11011010 RETL @0b11110010 RETL @0b01100110 RETL @0b10110110 RETL @0b10111110 RETL @0b11100000 RETL @0b11111110 RETL @0b11110110	

(*Continuation*)

	MAIN: • • • MOV A,0x10 ;*Get content of Register 0x10* CALL TRANS MOV 0x6,A ;*Output to 7-segment LED* • • • **NOTE** *Putting "@" in front of a word, indicates a literal.*

■ CONTR (Move CONT to A Register)

Syntax	CONTR
Operation	CONT → A
Status Affected	None
Description	Move data from the CONT register to the A register

Example	CONTR

■ IOR (Move IOCR to A Register)

Syntax	IOR R
Operation	IOCR → A
Status Affected	None
Description	Move data from control register IOCR to the A register

Example	IOR 0xf ;***Get the contents of IOCF*** MOV 0x10,A ;*Save in 0x10 register*

■ MOV (Move Data)

Syntax	MOV R,A
Operation	A → R
Status Affected	None
Description	Move the contents of the A register to R

Syntax	MOV A,R
Operation	R → A
Status Affected	Z
Description	Move the contents of R to the A register. If the result is "0", Z flag will be set. Otherwise, Z flag will be cleared.

Syntax	MOV R,R
Operation	R → R
Status Affected	Z
Description	Move the contents of R to R.

Syntax	MOV A,k
Operation	k → A
Status Affected	None
Description	Load immediate data (8-bit literal) into the A register.

Example	`Move data from accumulator to register:`
	`  MOV A,@0x11  ;Move immediate data to A`
	`  MOV 0x10,A  ;Move data from A to R`
	`  MOV A,9  ;Move data from R to A`
	`  MOV 0x10,0x10  ;Z flag is 0`

■ CLRA (Clear the A Register)

Syntax	CLRA
Operation	0 → A
Status Affected	1 → Z
Description	Reset Accumulator. Z flag is set

Example	`CLRA  ;Reset A. Z flag will be set`

■ CLR (Clear Register)

Syntax	CLR R
Operation	0 → R
Status Affected	1 → Z
Description	Reset register R. Z flag is set.

Example	`CLR 0x10  ;Reset register 0x10`

■ SUB (Subtract)

Syntax	SUB A,R
Operation	R - A → A
Status Affected	Z,C,DC
Description	Subtract the A register from R register. The result is placed in the A register.

Syntax	SUB R,A
Operation	R - A → R
Status Affected	Z,C,DC
Description	Subtract the A register from R register. The result is placed in the R register.

Syntax	SUB A,k
Operation	k - A → A
Status Affected	Z,C,DC
Description	Subtract the A register from the immediate data “k”. The result is placed in the A register.

Example

```
The following codes illustrate how to obtain
A = 0x99-0x55:
  MOV  A,@0x99
  MOV  0x10,A     ;R10 = 0x99
  MOV  A,@0x55
  SUB  A,0x10     ;A = 0x44
The following codes illustrate how to obtain
A = 0x02 – A:
  MOV  A,@0x01    ;A = 0x01
  SUB  A,@0x02    ;A = 0x02 - 0x01 = 0x01
                  ;C flag = 1 , result is
                  ;positive
```

■ DECA (Decrement R, Place in A)

Syntax	DECA R
Operation	R - 1 → A
Status Affected	Z
Description	Decrease the R register. The result is placed in A register.

Example

```
The following codes illustrate how to make a
16 - iteration loop:
STATUS == 3      ;Status Register
Z_FLAG == 2
  MOV  A,@0x10
  MOV  0x10,A
LOOP:
  DECA 0x10
  MOV  0x10,A
  JBS  STATUS,Z_FLAG
  JMP  LOOP
```

■ DEC (Decrement R)

Syntax	DEC R
Operation	R - 1 → R
Status Affected	Z
Description	Decrease the R register.

Example	The following codes illustrate how to make a 16 - Iteration loop: STATUS == 3 ;*Status Register* Z_FLAG == 2 MOV A,@0x10 MOV 0x10,A LOOP: DEC 0x10 JBS STATUS,Z_FLAG JMP LOOP

■ OR (Inclusive OR)

Syntax	OR A,R
Operation	A ∨ R → A
Status Affected	Z
Description	Inclusive OR the contents of A register with the R register. The result is placed in the A register.

Syntax	OR R,A
Operation	A ∨ R → R
Status Affected	Z
Description	Inclusive OR the contents of the A register with R register. The result is placed in the R register

Syntax	OR A,k
Operation	A ∨ k → A
Status Affected	Z
Description	Inclusive OR the contents of the A register with the immediate data specified by k. The result is placed in the A register

Example	The following codes illustrate how to obtain A = 0x55 + 0xAA = 0xFF: MOV A,@0x55 MOV 0x10,A ;*R10 = 0x55 + 0xAA* MOV A,@0xAA OR A,0x10 ;*A = 0xAA + 0x55 = 0xFF,* ;*Z flag = 0* or MOV A,@0x55 OR A,@0xAA ;*k = 0xAA, A = 0x55 + 0xAA =* ;*0xFF, Z flag = 0*

(*Continuation*)

	The following codes illustrate how to obtain R10 = 0x55 + 0xAA = 0xFF: MOV A,@0x55 MOV 0x10,A ;*R10 = 0x55* MOV A,@0xAA OR 0x10,A ;*R10 = 0x55 + 0xAA = 0xFF,* ;*Z flag = 0*

■ AND (And)

Syntax	AND A,R
Operation	A & R → A
Status Affected	Z
Description	AND the contents of the A register with the R register. The result is placed in the A register.

Syntax	AND R,A
Operation	A & R → R
Status Affected	Z
Description	AND the contents of the A register with the R register. The result is placed in R register.

Syntax	AND A,K
Operation	A & K → A
Status Affected	Z
Description	AND the contents of the A register with the immediate data specified by K. The result is placed in the A register.

Example	AND Port 6 with the R***10*** register, then output the result to Port 6: MOV A,0x6 ;*Input from Port 6* AND A,0x10 ;*AND with R**10*** MOV 0x6,A ;*Output to Port 6* R***10*** = R***11*** AND R*12* MOV A,0x11 MOV 0x10,A MOV A,0x12 AND 0x10,A ;*R**10** = R**11** AND R**12***

■ XOR (Exclusive OR)

Syntax	XOR A,R
Operation	A ⊕ R → A
Status Affected	Z
Description	The contents of the A register are XOR'ed with the R register. The result is placed in the A register.

Syntax	XOR R,A
Operation	A ⊕ R → R
Status Affected	Z
Description	Exclusive OR the A register with the R register. The result is placed in R register.

Syntax	XOR A,k
Operation	A ⊕ k → A
Status Affected	Z
Description	Exclusive OR the A register with the immediate data k. The result is placed in the A register.

Example	Check whether the content of Register 0x10 is 0x55. If not, jumps to ERROR subroutine: STATUS == 3 Z_FLAG == 2 MOV A,@0x55 XOR A,0x10 JBS STATUS,Z_FLAG JMP ERROR The following codes get R10 = R11 XOR R12: MOV A,0x11 MOV 0x10,A MOV A,0x12 XOR 0x10,A ;R10 = R11 XOR R12 The following codes get A = A XOR 0xF0: MOV A,@0x00 ;A = 0x00 XOR A,@0xF0 ;A = 0xF0

■ ADD (Add)

Syntax	ADD A,R
Operation	A + R → A
Status Affected	Z,C,DC
Description	The contents of the A register are added to the R register. The result is placed in the A register.

Syntax	ADD R,A
Operation	A + R → R
Status Affected	Z,C,DC
Description	ADD the contents of the A register to the R register. The result is placed in register R.

Syntax	ADD A,k
Operation	K + A → A
Status Affected	Z,C,DC
Description	The contents of the A register are added with the immediate data "k". The result is placed in the A register.

Example	The following codes get A = R11 + R12: MOV A,0x11 ADD A,0x12 ;A = R11 + R12 The following codes get R10 = R11 + R12: MOV A,0x11 MOV 0x10,A MOV A,0x12 ADD 0x10,A ;R10 = R11 + R12 The following codes get A = 0x01 + 0x01: MOV A,@0x01 ;A = 0x01 ADD A,@0x01 ;A = 0x02

■ COMA (Complement R, Place in A)

Syntax	COMA R
Operation	$\overline{R} \rightarrow A$
Status Affected	Z
Description	The contents of the R register are complemented. The result is placed in the A register.

Example	Input from Port 6, complement the contents; then output the result to Port 6: MOV A,0x6 MOV 0x10,A COMA 0x10 MOV 0x6,A

■ COM (Complement R)

Syntax	COM R
Operation	$\overline{R} \rightarrow R$
Status Affected	Z
Description	The contents of the R register are complemented. The result is placed in register R.

Example	The following codes complement the contents of R10: MOV A,@0x11 MOV 0x10,A ;R10 = 0x11 COM 0x10 ;R10 = 0xEE

■ INCA (Increment R, Place in the A Register)

Syntax	INCA R
Operation	R + 1 → A
Status Affected	Z
Description	The contents of the R register are incremented. The result is placed in the A register.

Example

```
Counts the external interrupt.  When count is
up to 100 times, some flags are set:
STATUS   == 0x03     ;Status Register
ISR      == 0x0f     ;Interrupt Status Register
FLAG     == 0x10
COUNTER  == 0x11
A_BUF    == 0x12     ;Buffer of A
S_BUF    == 0x13     ;Buffer of Status Register
TMO_FLG  == 0        ;Time out flag
TCIF     == 0        ;TCC interrupt flag
EXIF     == 3        ;External interrupt flag
Z_FLAG   == 2
EXT_INT
   DISI              ;Disable interrupt
   BC    ISR,EXIF    ;Reset EXIF flag
   MOV   A_BUF,A     ;Backup A
   MOV   A,STATUS
   MOV   S_BUF,A     ;Backup Status Register
   INCA  COUNTER     ;Increment count
   MOV   COUNTER,A
   XOR   A,@100      ;If counts to 100, set
                     ;TMO_FLG
   JBC   STATUS,Z_FLAG
   BS    FLAG,TMO_FLG
   MOV   A,S_BUF
   MOV   STATUS,A    ;Restore Status Register
   MOV   A,A_BUF     ;Restore A.  C flag in
                     ;STATUS is affected
   BC    STATUS,Z_FLG
   JBC   S_BUF,Z_FLG  ;Obtain the REAL C flag
   BS    STATUS,Z_FLG
   RTI
```

■ INC (Increment R)

Syntax	INC R
Operation	R + 1 → R
Status Affected	Z
Description	The contents of the R register are incremented.

Example

```
MOV A,@0x11
MOV     0x10,A       ;R10 = 0x11
INC     0x10         ;R10 = 0x12
```

■ DJZA (Decrement R, Place in the A Register, Skip if "0")

Syntax	DJZA R
Operation	R – 1 → A, skip if "0"
Status Affected	None
Description	Decrease the contents of the R register. The result is placed in the A register. If the result is "0", the next instruction which is already fetched is discarded.

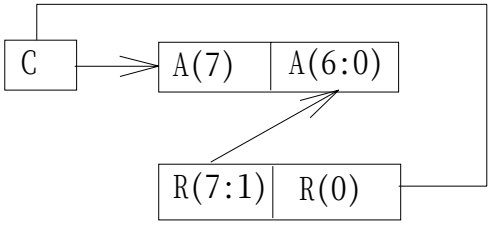
Example	HERE: DJZA 0x9 CONT: MOV A,0x10 SKIP: ADD A,@10 Before Instruction PC = address HERE After Instruction A=R9-1 if A = 0, PC = address SKIP if A ≠ 0, PC = address CONT

■ DJZ (Decrement R, Skip if "0")

Syntax	DJZ R
Operation	R – 1 → R, skip if "0"
Status Affected	None
Description	The contents of the R register are decreased. The result is placed in register R. If the result is "0", the next instruction which is already fetched is discarded.

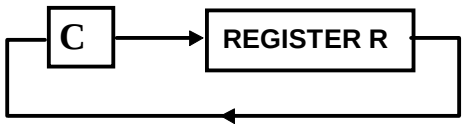
Example	MOV A,@100 MOV 0x10,A LOOP: • • • DJZ 0x10 *;Decrement R**10**. If the result* *;is not "0", executes the JMP* *;instruction, else skip the JMP* *;instruction.* JMP LOOP

■ RRCA (Rotate Right R through Carry, Place in the A Register)

Syntax	RRCA R
Operation	R(n) → A(n-1), R(0) → C, C → A(7)
Status Affected	C
Description	The contents of the R register are rotated 1-bit to the right through the Carry Flag. The result is placed in the A register. 

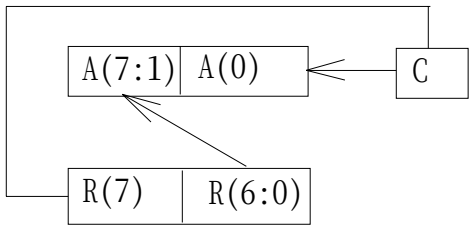
Example	The following codes perform how to rotate a register to the right (not to include C flag) and output the result to Port 5: BIT_BUF == 10 COUNTER == 11 MOV A,@0x8 MOV COUNTER,A LOOP: RRCA BIT_BUF ;Shift the LSB to C RRCA BIT_BUF ;Start right rotation MOV 0x5,A MOV BIT_BUF,A DJZ COUNTER JMP LOOP

■ RRC (Rotate Right R through Carry)

Syntax	RRC R
Operation	R(n) → R(n-1), R(0) → C, C → R(7)
Status Affected	C
Description	The contents of the R register are rotated 1-bit to the right through the Carry Flag. The result is placed in the R register. 

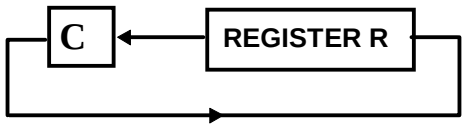
Example	MOV A,@0x0f CLR 0x3,0 ;Clear C flag MOV 0x10,A ;R10 = 00001111 RRC 0x10 ;R10 = 00000111, C = 1

■ RLCA (Rotate Left R through Carry, Place in the A Register)

Syntax	RLCA R
Operation	R(n) → A(n+1), R(7) → C, C → A(0)
Status Affected	C
Description	The contents of the R register are rotated 1-bit to the left through the Carry Flag. The result is placed in the A register. 

Example	Perform a rotate left operation of a 16-bit register (R***10***, R***11***):
	`RLCA 0x11 ;Shift the MSB of R11 into ;C flag`
	`RLC 0x10`
	`RLC 0x11`

■ RLC (Rotate Left R through Carry)

Syntax	RLC R
Operation	R(n) → R(n+1), R(7) → C, C → R(0)
Status Affected	C
Description	The contents of the R register are rotated 1-bit to the left through the Carry Flag. The result is placed in the R register. 

■ SWAPA (Swap R, Place in the A Register)

Syntax	SWAPA R
Operation	R(3 : 0) → A(7 : 4) R(7 : 4) → A(3 : 0)
Status Affected	None
Description	The upper and lower nibbles of register R are exchanged. The result is placed in the A register.

Example	Swap the contents of Port 6:
	`MOV A,0x6`
	`MOV 0x10,A`
	`SWAP 0x10`
	`MOV 0x6,A`

■ SWAP (Swap R)

Syntax	SWAP R
Operation	R(3 : 0) ⇔ R(7 : 4)
Status Affected	None
Description	The upper and lower nibbles of Register R are exchanged.

Example	`MOV A,@0x43`
	`MOV 0x10,A ;R10 = 0x43`
	`SWAP 0x10 ;R10 = 0x34`

■ JZA (Increment R, Place in the A Register, Skip if "0")

Syntax	JZA R
Operation	R + 1 → A, skip if result = "0"
Status Affected	None
Description	The contents of the R register are incremented. The result is placed in the A register. If the result is "0", the next instruction which is already fetched is discarded.

Example	`Port 6 outputs incremental binary signal:`
	`  MOV A,@x00`
	`LOOP:`
	`  MOV 0x6,A`
	`  MOV 0x10,A`
	`  JZA 0x10`
	`  JMP LOOP`

■ JZ (Increment R, Skip if "0")

Syntax	JZ R
Operation	R + 1 → R, skip if result = "0"
Status Affected	None
Description	Increase the contents of the R register. The result is placed in the R register. If the result is "0", the next instruction which is already fetched is discarded.

Example	`HERE:`
	`  JZ 0x10`
	`CONT:`
	`  MOV A,0x10`
	`SKIP:`
	`  ADD A,@10`
	`Before Instruction`
	`PC = address HERE`
	`After Instruction`
	`R10 = R10-1`
	`if R10 = 0, PC = address SKIP`
	`if R10 ≠ 0, PC = address CONT`

■ BC (Bit Clear)

Syntax	BC R,b
Operation	0 → R(b)
Status Affected	None
Description	Bit “b” in Register R is reset.

Example	`MOV A,@0x0f`
	`MOV 0x10,A ;R10 = 00001111`
	`BC 0x10,3 ;R10 = 00000111`

■ BS (Bit Set)

Syntax	BS R,b
Operation	1 → R(b)
Status Affected	None
Description	Bit “b” in register R is set.

Example	`Set the C flag in the Status Register:` `BS 0x3,2`

■ JBC (Bit Test, Skip if Clear)

Syntax	JBC R,b
Operation	if R(b) = “0”, skip
Status Affected	None
Description	If bit “b” in Register R is “0”, then the next instruction is skipped.

Example	`Test the contents of R10. If “0”, Port 5.0 outputs “0”, else Port 5.0 outputs “1”:`
	`JBC 0x10,0`
	`BS 0x5,0`
	`JBS 0x10,0`
	`BC 0x5,0`

■ JBS (Bit Test, Skip if Set)

Syntax	JBS R,b
Operation	if R(b) = 1, skip
Status Affected	None
Description	If bit “b” in register R is “1”, then the next instruction is skipped.

Example	HERE JBC 0x9,3 CONT MOV A,@10 SKIP ADD A,0x10 Before Instruction: PC = address HERE After Instruction: if R9(3) = 0, PC = address CONT if R9(3) ≠ 0, PC = address SKIP

■ CALL (Subroutine Call within one ROM Page)

Syntax	CALL k
Operation	PC + 1 → [Top of Stack] k → PC(9 : 0) R3(7 : 5) → PC(12 : 10)
Status Affected	None
Description	When invoking a subroutine call, the return address is pushed into the top of the stack first. Then the 10-bit address specified by "k" is loaded into PC (9 : 0). The Page Select Bits PS2, PS1, & PS0 (in R3, Status Register) are loaded into PC (12 : 10). The CALL instruction can only call program subroutine within one ROM page.

Example	HERE: CALL SUBRTN CONT: MOV A,@10 Before Instruction PC = address HERE After Instruction PC = address SUBRTN [Top of Stack] = address CONT

■ JMP (Unconditional Branch)

Syntax	JMP k
Operation	k → PC(9 : 0) R3(7 : 5) → PC(12 : 10)
Status Affected	None
Description	When invoking an unconditional jump, the 10-bit address specified by "k" is loaded into PC (9 : 0). The Page Select Bits PS2, PS1, & PS0 (in R3, Status Register) are loaded into PC (12 : 10). The JMP instruction can only jump a PC address within one ROM page.

Example	HERE JMP BRANCH Before Instruction PC = address HERE After Instruction PC = address BRANCH **NOTE** *Both PC addresses of HERE and BRANCH are within the same ROM page.*

■ INT (Software Interrupt)

Syntax	INT
Operation	PC + 1 → [Top of Stack] 0001H → PC
Status Affected	None
Description	Invoke an interrupt subroutine. The return address (PC+1) is pushed into the top of the stack. The PC is set to 0x0001.

Example	ORG 0x001 JMP SET_INT • • SET_INT: • • RETI MAIN: • • • HERE INT CONT CLRA Before Instruction PC = address HERE After Instruction PC = 0001H [Top of Stack] = address CONT

■ PAGE (Switch PAGE of ROM)

Syntax	PAGE k
Operation	k → R3(6:5)
Status Affected	None
Description	The page instruction is equal to the page select bits PS2, PS1, & PS0 (in R3, Status Register) which are loaded into PC (12 : 10).

Example

```
   ORG 0x100
   JMP BRANCH_1   ;"JMP BRANCH_1" instruction
                  ;is within one ROM page
   ORG 0x200
BRANCH_1:
   •
   •
   PAGE 1
   JMP BRANCH_2   ;"JMP BRANCH_2" instruction
                  is out of one ROM page,
                  therefore, it needs to add
                  extra instruction "PAGE k" to
                  switch to another ROM page
                  because the address of
                  BRANCH_2 is located at second
                  ROM page (0x400~0x7FF). The
                  value of k is set to 1.
   •
   ORG 0x500
BRANCH_2:
   •
   •
```

■ LCALL (Subroutine Call)

Syntax	LCALL k
Operation	PC + 1 → [Top of Stack] k → PC(12 : 0)
Status Affected	None
Description	When invoking a subroutine call, the return address is pushed into the top of the stack first. Then the 13-bit address specified by "k" is loaded into PC (12 : 0). The LCALL instruction can call any program subroutine without the within one ROM page restriction.

Example	HERE: LCALLSUBRTN CONT: MOV A,@10 Before Instruction PC = address HERE After Instruction PC = address SUBRTN [Top of Stack] = address CONT **NOTE** *The program subroutine SUBRTN can call any PC address without the within one ROM page restriction.*

■ LJMP (Unconditional Branch)

Syntax	LJMP k
Operation	k → PC(12 : 0)
Status Affected	None
Description	When invoking an unconditional jump, the 13-bit address specified by “k” is loaded into PC (12 : 0). The LJMP instruction can jump any PC address without the within one ROM page restriction.

Example	HERE LJMP BRANCH Before Instruction PC = address HERE After Instruction PC = address BRANCH **NOTE** *Both PC addresses of HERE and BRANCH can be located anywhere.*

■ TBRD (Table Read)

Syntax	TBRD R
Operation	ROM[(TABPTR)] → R
Status Affected	None
Description	The TBPTL (low byte of ROM address pointer) and TBPTH (high byte of ROM address pointer) registers which are used to point to the ROM address; are set separately by switched HLB bit. Then TBRD instruction is executed to obtain the corresponding ROM code and put the code to the assigned Register R. The two partial code data are combined into one complete ROM code.

Example

```
CLR TBPTL
CLR TBPTH      ;Point to ROM Address 0x0
BC HLB         ;Obtain low byte of ROM code
TBRD 0x10      ;Place the low byte of ROM code
               ;to Register 0x10
BS HLB         ;Obtain high five bits of ROM
               ;code
TBRD 0x11
MOV A,@0x1F
AND 0x11, A    ;Place the low byte of ROM code
               ;to Register 0x11
```