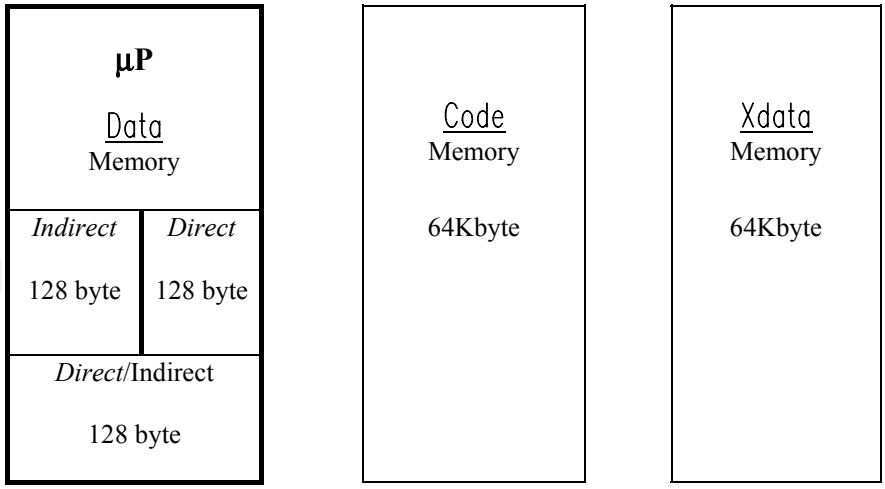
µP 8051/80535 INSTRUCTION SET

Jacek Majewski , 1994

µP Algorithm

1. Fetch from memory$_{Code}$ instruction:
   Instr. Reg. ←(PC)$_{Code}$
2. PC ←PC+ No_of_bytes
3. Operation
4. Go to 1.

µP Memory Maps



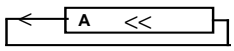
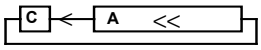
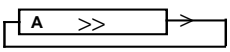
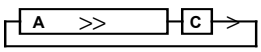
Notes on Data *addr*essing modes:

Rn	working register R0-R7
direct	128 internal RAM locations, any l/O port, control or status register located in Data memory space
@R0/1	in*direct* Data memory space (internal or external RAM) location *addr*essed by register R0 or R1
# *data*	8-*bit* constant included in instruction
#*data* 16	16-*bit* constant included as bytes 2 and 3 of instruction
bit	8-*bit addr*ess of µP *bit*s located in Data memory space
A	accumulator
( )	contents of Data memory space (internal data RAM of µP); description ( ) is equivalent to ( )$_{Data}$
( )$_{Code}$	contents of Code program memory space (internal/external ROM of µP)
( )$_{Xdata}$	contents of Xdata external Data memory space (external Data RAM of µP)
C*addr* $_{16}$	destination *addr*ess for LCALL and LJMP may be anywhere within the 64 Kbyte program memory Code *addr*ess space
C*addr* $_{11}$	destination *addr*ess for ACALL and AJMP will be within the same 2 Kbyte page of program memory as the first byte of the following instruction.
C*rel*	SJMP and all conditional jumps include an 8-*bit* offset byte. Range is + 127/-128 bytes *rel*ative to first byte of the following instruction. Value of C*rel* is sign-extended to double bytes before e addition to PC.
sign	sign extension
nosgn	no sign extension
←	move
↔	exchange
⊥	concatenate

C language operators:

+ - * / %	add, subtrac, multiply, divide, moduilo
& | ^ ~	and, or, xor, not (bitwise operators)

Mnemonic	Operation	Code	M Cycles
Arithmetic operations			
ADD A, Rn	A ←A + Rn flags : CY OV AC	0010 1rrr	1
ADD A, *direct*	A ←A + (*direct*) flags : CY OV AC	25 *direct*	1
ADD A, @R0/1	A ←A + (R0/1) flags : CY OV AC	26 /27	1
ADD A, #*data*	A ←A + *data* flags : CY OV AC	24 *data*	1
ADDC A, Rn	A ←A + Rn +CY flags : CY OV AC	0011 1rrr	1
ADDC A, *direct*	A ←A + (*direct*) + CY flags : CY OV AC	35 *direct*	1
ADDC A, @R0/1	A ←A + (R0/1) + CY flags : CY OV AC	36 /37	1
ADDC A, #*data*	A ←A + *data* + CY flags : CY OV AC	34 *data*	1
SUBB A, Rn	A ←A – Rn – CY flags : CY OV AC	1001 1rrr	1
SUBB A, *direct*	A ←A – (*direct*) – CY flags : CY OV AC	95 *direct*	1
SUBB A, @R0/1	A ←A – (R0/1) – CY flags : CY OV AC	96 /97	1
SUBB A, #*data*	A ←A – *data* – CY flags : CY OV AC	94 *data*	1
INC A	A ←A + 1	04	1
INC Rn	Rn ←Rn + 1	0000 1rrr	1
INC *direct*	(*direct*) ←(*direct*) + 1	05 *direct*	1
INC @R0/1	(R0/1) ←(R0/1) + 1	06 /07	1
DEC A	A ←A – 1	14	1
DEC Rn	Rn ←Rn – 1	0001 1rrr	1
DEC *direct*	(*direct*) ←(*direct*) – 1	15 *direct*	1
DEC @R0/1	(R0/1) ←(R0/1) – 1	16 /17	1
INC DPTR	DPTR ← DPTR + 1	A3	2
MUL AB	B ⊥ A ← A * B flags : CY=0 OV	A4	4
DIV AB	A ← A / B (result) A ← A % B (remainder) flags : CY=0 OV	84	4
DA A	if A_{3-0} >0 or AC=1 : A_{3-0}← A_{3-0} +6 if A_{7-4} >0 or CY=1 : A_{7-4}← A_{7-4} +6 flag : CY	D4	1
Logical operation			
ANL A, Rn	A ← A & Rn	0101 1rrr	1
ANL A, *direct*	A ← A & (*direct*)	55 *direct*	1
ANL A, @R0/1	A ← A & (R0/1)	56 /57	1
ANL A, #*data*	A ← A & *data*	54 *data*	1

Mnemonic	Operation	Code	M Cycles
ANL *direct*, A	(*direct*) ← (*direct*) & A	**52** *direct*	1
ANL *direct*, *#data*	(*direct*) ← (*direct*) & *data*	**53** *direct data*	2
ORL A, Rn	A ←A | Rn	**0100 1*rrr***	1
ORL A, *direct*	A ←A | (*direct*)	**45** *direct*	1
ORL A, @R0/1	A ←A | (R0/1)	**46 /47**	1
ORL A, *#data*	A ←A | *data*	**44** *data*	1
ORL *direct*, A	(*direct*) ← (*direct*) | A	**42** *direct*	1
ORL *direct*, *#data*	(*direct*) ← (*direct*) | *data*	**43** *direct data*	2
XRL A, Rn	A ←A ^ Rn (^ XOR)	**0110 1*rrr***	1
XRL A, *direct*	A ←A ^ (*direct*)	**65** *direct*	1
XRL A, @R0/1	A ←A ^ (R0/1)	**66 /67**	1
XRL A, *#data*	A ←A ^ *data*	**64** *data*	1
XRL *direct*, A	(*direct*) ← (*direct*) ^ A	**62** *direct*	1
XRL *direct*, *#data*	(*direct*) ← (*direct*) ^ *data*	**63** *direct data*	2
CLR A	A ← 0	**E4**	1
CPL A	A ← ~A	**F4**	1
RL A		**23**	1
RLC A		**33**	1
RR A		**03**	1
RRC A		**13**	1
SWAP A	$A_{7-4} \leftrightarrow A_{3-0}$	**C4**	1

Data transfers

MOV A, Rn	A ← Rn	**1110 1*rrr***	1
MOV A, *direct* ~~MOV A, ACC~~	A ← (*direct*)	**E5** *direct*	1
MOV A, @R0/1	A ← (R0/1)	**E6 /E7**	1
MOV A, *#data*	A ← *data*	**74** *data*	1
MOV Rn, A	Rn ← A	**1111 1*rrr***	1
MOV Rn, *direct*	Rn ← (*direct*)	**A8** *direct*	2
MOV Rn, *#data*	Rn ← *data*	**0111 1*rrr*** *data*	1
MOV *direct*, A	(*direct*) ← A	**F5** *direct*	1
MOV *direct*, Rn	(*direct*) ← Rn	**88** *direct*	2

Mnemonic	Operation	Code	M Cycles
MOV $direct_d$, $direct_s$	$(direct_d) \leftarrow (direct_s)$	**85** $direct_s$ $direct_d$	2
MOV *direct*, @R0/1	$(direct) \leftarrow (R0/1)$	**86** *direct* **87** *direct*	2
MOV *direct*, #*data*	$(direct) \leftarrow data$	**75** *direct data*	2
MOV @R0/1, A	$(R0/1) \leftarrow A$	**F6 /F7**	1
MOV @R0/1, *direct*	$(R0/1) \leftarrow (direct)$	**A6** *direct*	2
MOV @R0/1, #*data*	$(R0/1) \leftarrow data$	**76** *data* **77** *data*	1
MOV DPTR, #$data_{16}$	DPTR $\leftarrow data_{16}$	**90** $data_{16}$ (HiLo)	2
MOVC A, @A+DPTR	$A \leftarrow (A_{nosign}+DPTR)_{Code}$	**93**	2
MOVC A, @A+PC	$A \leftarrow (A_{nosign}+PC)_{Code}$	**83**	2
MOVX A, @R0/1	$A \leftarrow (P2 \perp R0/1)_{Xdata}$	**E2 /E3**	2
MOVX A, @DPTR	$A \leftarrow (DPTR)_{Xdata}$	**E0**	2
MOVX @R0/1, A	$(P2 \perp R0/1)_{Xdata} \leftarrow A$	**F2 /F3**	2
MOVX @DPTR, A	$(DPTR)_{Xdata} \leftarrow A$	**F0**	2
PUSH *direct*	SP$\leftarrow$ SP+1; $(SP) \leftarrow (direct)$	**C0** *direct*	2
POP *direct*	$(direct) \leftarrow (SP)$; SP$\leftarrow$ SP–1	**D0** *direct*	2
XCH A, Rn	$A \leftrightarrow Rn$	**1100 1rrr**	1
XCH A, *direct*	$A \leftrightarrow (direct)$	**C5** *direct*	1
XCH A, @R0/1	$A \leftrightarrow (R0/1)$	**C6 /C7**	1
XCHD A, @R0/1	$A_{3-0} \leftrightarrow (R0/1)_{3-0}$	**D6 /D7**	1
Program and machine control			
ACALL C$addr_{11}$ 2S	SP$\leftarrow$ SP+2; $(SP : SP-1) \leftarrow PC$ PC$\leftarrow PC_{15-11} \perp C addr_{10-0}$	**aaa1 0001** C$addr_{7-0}$	2
LCALL C$addr_{16}$ 2	SP$\leftarrow$ SP+2; $(SP : SP-1) \leftarrow PC$ PC$\leftarrow C addr_{15-0}$	**12** C$addr_{16}$ (HiLo)	2
RET	PC$\leftarrow$(SP : SP–1);SP $\leftarrow$ SP–2_	**22**	2
RETI	PC$\leftarrow$(SP : SP–1);SP $\leftarrow$ SP–2_	**32**	2
AJMP C$addr_{11}$	PC$\leftarrow PC_{15-11} \perp C addr_{10-0}$	**aaa0 0001** C$addr_{7-0}$	2
LJMP C$addr_{16}$	PC$\leftarrow C addr_{15-0}$	**02** C$addr_{16}$ (HiLo)	2
SJMP C*rel*	PC$\leftarrow$ PC + Crel_{sign}	**80** C*rel*	2
JMP @A+DPTR	PC$\leftarrow A_{nosign}$+DPTR	**73**	2
JZ C*rel*	if A=0 : PC $\leftarrow$ PC + Crel_{sign}	**60** C*rel*	2
JNZ C*rel*	if A$\neq$0 : PC $\leftarrow$ PC + Crel_{sign}	**70** C*rel*	2
JC C*rel*	if CY=1 : PC $\leftarrow$ PC + Crel_{sign}	**40** C*rel*	2

Mnemonic	Operation	Code	M Cycles
JNC *Crel*	if CY=0 : PC ← PC + $Crel_{sign}$	**50** *Crel*	2
JB *bit*, *Crel*	if *bit*=1 : PC ← PC + $Crel_{sign}$	**20** *bit Crel*	2
JNB *bit*, *Crel*	if *bit*=0 : PC ← PC + $Crel_{sign}$	**30** *bit Crel*	2
JBC *bit*, *Crel*	if *bit*=1 : *bit*←0 PC← PC + $Crel_{sign;}$	**10** *bit Crel*	2
CJNE A, *direct*, *Crel*	if A≠(*direct*) : PC← PC + $Crel_{sign}$ if A<(*direct*) : CY←1 else CY←0	**B5** *direct Crel*	2
CJNE A, #*data*, *Crel*	if A≠*data* : PC← PC + $Crel_{sign}$ if A<*data* : CY←1 else CY←0	**B4** *data Crel*	2
CJNE Rn, #*data*, *Crel*	if Rn≠*data* : PC← PC + $Crel_{sign}$ if Rn<*data* : CY←1 else CY←0	**B8** *data Crel*	2
CJNE @R0/1, #*data*, *Crel*	if (R0/1)≠*data* : PC← PC + $Crel_{sign}$ if (R0/1)<*data* : CY←1 else CY←0	**B6** *data Crel* **/** **B7** *data Crel*	2
DJNZ Rn, *Crel*	Rn ←Rn – 1 if Rn≠0 : PC← PC + $Crel_{sign}$	**1101 1rrr** *Crel*	2
DJNZ *direct*, *Crel*	(*direct*) ←(*direct*)– 1 if (*direct*)≠0 : PC← PC + $Crel_{sign}$	**D5** *direct Crel*	2
NOP	;	**00**	1

Bololean variable manipulation

CLR *bit*	*bit* ←0	**C2** *bit*	1
SETB C	CY ← 1	**D3**	1
SETB *bit*	*bit* ← 1	**D2** *bit*	1
CPL C	CY ← ~ CY	**B3**	1
CPL *bit*	*bit* ← ~ *bit*	**B2** *bit*	1
ANL C, *bit*	CY ← CY & *bit*	**82** *bit*	2
ANL C, /*bit*	CY ← CY & ~ *bit*	**B0** *bit*	2
ORL C, *bit*	CY ← CY | *bit*	**72** *bit*	2
ORL C, /*bit*	CY ← CY | ~ *bit*	**A0** *bit*	2
MOV C, *bit*	CY ← *bit*	**A2** *bit*	1
MOV *bit*, C	*bit* ← CY	**92** *bit*	2

Program Status Word register (PSW) D0 HEX.

CY	AC	F0	RS1	RS0	OV	F1	P
			Bank Register Select				
D7H	D6H	D5H	D4H	D3H	D2H	D1H	D0H