**Assembly coding ABC-2**

1. **51-architecture and Operating Modes**

|  |  |
| --- | --- |
| 51 architecture (stand-alone op.-mode) | 51 architecture (external-memory op.-mode) |
| data-space of 256B 4KB  code-space    REGs x 4-bank    Bit-map x 16B PC  User-space x  SPF x 128B decoder  **P0**  **P1**  data-path **P2**  address for machine-code fetch **P3**  machine-code **EA**  control-path | 4KB  code-space 64KB code-memory  AH AH  C\*/D\*  -PC **P2**  AL **P0**  decoder AL  data-bus  int.- data-space  ***MOVX***  ***MOV*** 64KB data-memory  **P1**  PSW  **P3**  **EA** |

|  |
| --- |
| Processor Status Word: PSW |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | Cy | AC | X | RS1 | RS0 | OV | X | P |   \* Cy : carry  \* AC: auxiliary carry  \* RS1:RS0: register-bank selector  \* OV: over-flow  \* P: odd-parity |
| internal data-memory structure: **Register Banks** |
| \* 32 bytes, address 0000H-001FH, arranged in 4 reg-bank  0000H R0 0008H R0 0010H R0 0018H R0  R1 R1 R1 R1  BANK0 . . . BANK1 . . . BANK2 . . . BANK3 . . .  R6 R6 R6 R6  R7 R7 R7 R7  \* selected by RS1-RS0 pair in PSW |
| internal data-memory structure: **User’s Space** |
| \* 80 bytes, address 30H-7FH |

|  |
| --- |
| internal data-memory structure: **Bit Map** |
| \* 16 bytes, address 20H-2FH  \* 128 its with bit-address 00H-07FH  20H   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 07H: |  |  |  |  |  |  | 00H: | | 0FH: |  |  |  |  |  |  | 08H: | | . . . | | | | | | | | | 7FH: |  |  |  |  |  |  | 78H: |   . . .  2FH |
| internal data-memory structure: **Special Function Registers** (**SPF**-area) |
| \* 128 bytes, address 80H-0FFH  **IO-ports** **stack pointer timer-control SIO-control interrupt**  P0\*: 80H SP: 81H TMOD: 89H SCON\*: 98H IE\*: 0A8H  P1\*: 90H **accumulator** TCON\*: 88H SBUF: 99H IP\*: 0B8H  P2\*: 0A0H A\*: 0E0H TH0: 8CH  P3\*: 0B0H TL0: 8AH  **data-pointer** **PSW** TH1: 8DH  DPL: 82H PSW\*: 0D0H TL1: 8BH  DPH: 83H PCON: 87H [SPF-Reg\*: bit-addressable]  **B-reg**  B\*: 0F0H |

1. **51 Instruction Set and Code Book**

|  |  |  |  |
| --- | --- | --- | --- |
| arithmetic/logic operations | | data movement/initialization | |
| pneumonic | machine code | pneumonic | machine code |
| ADD A, Rn 28H + n ; n: 0-7  ADD A, @Ri 26H + I ; i: 0-1  ADD A, diretc8  25H  ADD A, #dada8  24H  ADDC A, Rn 38H + n  ADDC A, @Ri  36H + i  ADDC A, diretc8 35H  ADDC A, #dada8 34H  SUBB A, Rn 98H + n  SUBB A, @Ri 96H + i  SUBB A, diretc8 95H  SUBB A, #dada8 94H  MUL A, B A4H  DIV A, B 84H  ANL A, Rn 58H + n  ANL A, @Ri 56H + i  ANL A, direct8 55H  ANL A, #data8 54H  ANL direct8, A 52H  ANL direct8, #data8 53H  ORL A, Rn 48H + n  ORL A, @Ri 46H + i  ORL A, direct8 45H  ORL A, #data8 44H  ORL direct8, A 42H  ORL direct8, #data8 43H  XRL A, Rn 68H + n  XRL A, @Ri 66H + i  XRL A, direct8 65H  XRL A, #data8 64H  XRL direct8, A 62H  XRL direct8, #data8 63H | | MOV A, Rn E8H + n  MOV A, @Ri E6H + i  MOV A, diretc8  E5H  MOV Rn, A F8H + n  MOV @Ri, A F6H + i  MOV diretc8, A F5H  MOV Rn, direct8 A8H + n  MOV @Ri, direct8 A6H  MOV direct8, Rn 88H + n  MOV direct8, @Ri 86H + i  MOV direct18, direct28 85H  MOV A, #dada8  74H  MOV Rn, #data8 78H + 8  MOV @Ri, #data8 76H + i  MOV direct8, #data8 75H  MOV DPTR16, #data16 90H  MOVC A, @A+PC16 83H  MOVC A, @A+DPTR 93H  MOVX A, @Ri E2H + i  MOVX A, @DPTR E0H  MOVX @Ri, A F2H + i  MOVX @DPTR, A F0H  XCH A, Rn C8H + n  XCH A, @Ri C6H + i  XCH A, direct8 C5H  XCHD A, @Ri D6H | |
| Accumulator operations | | increment/decrement operations | |
| DA A D4H  CLR A E4H  CPL A F4H  RL A 23H  RR A 03H  RLC A 33H  RRC A 13H  SWAP A C4H | | INC A 04H  INC Rn 08H + n  INC @Ri 06H + i  INC direct8 05H  INC DPTR16 A3H  DEC A 14H  DEC Rn 18H + n  DEC @Ri 16H + i  DEC direct8 15H | |
| BIT operations | | execution flow control | |
| CLR C C3H  CPL C B3H  SETB C D3H  CLR bit8 C2H  CPL bit8 B2H  SETB bit8 D2H  MOV C, bit8 A2H  MOV bit8, C 92H  ANL C, bit8 82H  ANL C, /bit8 B0H  ORL C, bit8 72H  ORL C, /bit8 A0H | | ACALL addr11 X1H ; X: odd value  LCALL addr16 12H  RET 22H  RETI 32H  AJMP addr11 X1H ; X: even value  LJMP addr16 02H  JMP @A+DPTR 73H  SJMP rel8 80H  JZ rel8  60H  JNZ rel8  70H  JC rel8  40H  JNC rel8  50H  JB bit8, rel8  20H  JNB bit8, rel8  30H  JBC bit8, rel8  10H  DJNZ Rn, rel8 D8H + n  DJNZ direct8, rel8 D5H  CJNZ A, direct8, rel8 B5H  CJNZ A, #data8, rel8 B4H  CJNZ Rn, #data8, rel8 B8H + n  CJNZ @Ri, #data8, rel8 B6H + i | |
| [syntax convention]  \* OPTR  OPTR OPND1  OPTR OPND1, OPND2  \* **#**data8: where *data8*is an 8-bit unsigned binary value to be set for opnd1 (the destination);  \* direct8: where *direct8*is an 8-bit unsigned binary value interpreted as the address of the OPND in an instruction;  \* Rn: general purpose registers R0-R7 residing in one of the 4 register banks (located from 00h to 1FH in the 256-byte built-in RAM);  \* @Ri: indexing register R0-R1, in which the content of R0/R1 is treated as address of the targeted OPND, instead of the OPND itself;  \* bit8: where *bit8*is an 8-bit unsigned binary value interpreted as the address of every individual bit in Bit-Map area, or of certain registers in SPF-area (the accumulator, A, for instance; | | | |

1. **51-instruction and Effects**

|  |  |  |  |
| --- | --- | --- | --- |
| arithmetic/logic operations | | data movement/initialization | |
| pneumonic | effects (PSW) | pneumonic | effects |
| ADD A, Rn A 🡨 A + Rn  ADD A, @Ri A 🡨 A + (Ri)  ADD A, diretc8  A 🡨 A + (direct8)  ADD A, #dada8  A 🡨 A + data8  ADDC A, Rn A 🡨 A + Rn + Cy  ADDC A, @Ri  A 🡨 A + (Ri) + Cy  ADDC A, diretc8 A 🡨 A + (direct8) + Cy  ADDC A, #dada8 A 🡨 A + data8 + Cy  SUBB A, Rn A 🡨 A - Rn  SUBB A, @Ri A 🡨 A – (Ri) - Cy  SUBB A, diretc8 A 🡨 A - (direct8) - Cy  SUBB A, #dada8 A 🡨 A - data8 - Cy  MUL A, B AB 🡨 A x B ; ???  DIV A, B A:Q 🡨 A ÷ B ; ???  B: R 🡨 A ÷ B ; ???  ANL A, Rn A 🡨 A ˆ Rn  ANL A, @Ri A 🡨 Aˆ (Ri)  ANL A, direct8 A 🡨 Aˆ (direct8)  ANL A, #data8 A 🡨 Aˆ data8  ANL direct8, A A 🡨 (direct8) ˆ A  ANL direct8, #data8 A 🡨 (direct8) ˆ data8  ORL A, Rn A 🡨 A ˇ Rn  ORL A, @Ri A 🡨 A ˇ (Ri)  ORL A, direct8 A 🡨 A ˇ (direct8)  ORL A, #data8 A 🡨 A ˇ data8  ORL direct8, A A 🡨 (direct8) ˇ A  ORL direct8, #data8 A 🡨 (direct8) ˇ data8  XRL A, Rn A 🡨 A ⊕ Rn  XRL A, @Ri A 🡨 A ⊕ (Ri)  XRL A, direct8 A 🡨 A ⊕ (direct8)  XRL A, #data8 A 🡨 A ⊕ data8  XRL direct8, A A 🡨 (direct8) ⊕ A  XRL direct8, #data8 A 🡨 (direct8) ⊕ data8 | | MOV A, Rn A 🡨 Rn  MOV A, @Ri A 🡨 (Ri)  MOV A, diretc8  A 🡨 (direct8)  MOV Rn, A Rn 🡨 A  MOV @Ri, A (Ri) 🡨 A  MOV diretc8, A (direct8) 🡨 A  MOV Rn, direct8 Rn 🡨 (direct8)  MOV @Ri, direct8 (Ri) 🡨 (direct8)  MOV direct8, Rn (direct8) 🡨 Rn  MOV direct8, @Ri (direct8) 🡨 (Ri)  MOV direct18, direct28 (direct18) 🡨 (direct28)  MOV A, #dada8  A 🡨 data8  MOV Rn, #data8 Rn 🡨 data8  MOV @Ri, #data8 (Ri) 🡨 data8  MOV direct8, #data8 (direct8) 🡨 data8  MOV DPTR16, #data16 DPTR 🡨 data16  **; code-space access**  MOVC A, @A+PC A 🡨 (A + PC)  MOVC A, @A+DPTR A 🡨 (A + DPTR)  **; external data-space access**  MOVX A, @Ri A 🡨 (Ri\*)  MOVX A, @DPTR A 🡨 (DPTR\*)  MOVX @Ri, A (Ri) 🡨 A  MOVX @DPTR, A (DPTR) 🡨 A  XCH A, Rn A 🡨 🡪 Rn  XCH A, @Ri A 🡨 🡪 (Ri)  XCH A, direct8 A 🡨 🡪 (direct8)  XCHD A, @Ri AL 🡨 🡪 (Ri)L | |
| Accumulator operations | | increment/decrement operations | |
| DA A Decimal Adjustment . . .  CLR A A 🡨 00H  CPL A A 🡨 /A  RL A A(j+1)mod8 🡨 Aj j: 0-7  RR A Aj 🡨 A(J+1)mod8 j: 0-7  RLC A ≡ RL CA where CA= Cy : A  RRC A ≡ RR AC where AC= A : Cy  SWAP A AH 🡨🡪 AL | | INC A A 🡨 A++  INC Rn Rn 🡨 Rn++  INC @Ri (Ri) 🡨 (Ri)++  INC direct8 (direct8) 🡨 (direct8)++  INC DPTR16 DPTR 🡨 DPTR++  DEC A A 🡨 A--  DEC Rn Rn 🡨 Rn--  DEC @Ri (Ri) 🡨 (Ri)--  DEC direct8 (direct8) 🡨 (direct8)-- | |
| BIT operations | | execution flow control | |
| CLR C Cy 🡨 0  CPL C Cy 🡨 /Cy  SETB C Cy 🡨 1  CLR bit8 (bit8) 🡨 0  CPL bit8 (bit8) 🡨 /(bit8)  SETB bit8 (bit8) 🡨 1  MOV C, bit8 Cy 🡨 (bit8)  MOV bit8, C (bit8) 🡨 Cy  ANL C, bit8 Cy 🡨 Cy ˆ (bit8)  ANL C, /bit8 Cy 🡨 Cy ˆ /(bit8)  ORL C, bit8 Cy 🡨 Cy ˇ (bit8)  ORL C, /bit8 Cy 🡨 Cy ˇ /(bit8) | | AJMP addr11 PC 🡨 addr11  LJMP addr16 PC 🡨 addr16  JMP @A+DPTR PC 🡨 A+DPTR  SJMP rel8 PC 🡨 PC + rel8  JZ rel8  if(ZERO) PC 🡨 PC + rel8  else as is  JNZ rel8  if(nonZERO) PC 🡨 PC + rel8  else as is  JC rel8  if(Cy) PC 🡨 PC + rel8  else as is  JNC rel8  if(/Cy) PC 🡨 PC + rel8  else as is  JB bit8, rel8  if((bit8)) PC 🡨 PC + rel8  else as is  JNB bit8, rel8  if(/(bit8)) PC 🡨 PC + rel8  else as is  JBC bit8, rel8  if((bit8)) PC 🡨 PC + rel8  else as is  (bit8) 🡨 1  DJNZ Rn, rel8  Rn- -  if(Rn: nonZERO) PC 🡨 PC + rel8  else as is  DJNZ direct8, rel8  (direct8)- -  if((direct8): nonZERO) PC 🡨 PC + rel8  else as is  CJNZ A, direct8, rel8  if(A!=(direct8)) PC 🡨 PC + rel8  else as is  CJNZ A, #data8, rel8  if(A!=data8) PC 🡨 PC + rel8  else as is  CJNZ Rn, #data8, rel8  if(Rn!=data8) PC 🡨 PC + rel8  else as is  CJNZ @Ri, #data8, rel8  if((Ri)!=data8) PC 🡨 PC + rel8  else as is  ACALL addr11 SP++  (SP) 🡨 PCL  SP++  (SP) 🡨 PCH  PC 🡨 addr11  LCALL addr16 SP++  (SP) 🡨 PCL  SP++  (SP) 🡨 PCH  PC 🡨 addr16  RET PCH 🡨 (SP)  SP - -  PCL 🡨 (SP)  SP - -  RETI PCH 🡨 (SP)  SP - -  PCL 🡨 (SP)  SP - -  Interrupt-at-service cleared | |
| [syntax convention]  \* OPTR  OPTR OPND1  OPTR OPND1, OPND2  \* **#**data8: where *data8*is an 8-bit unsigned binary value to be set for opnd1 (the destination);  \* direct8: where *direct8*is an 8-bit unsigned binary value interpreted as the address of the OPND in an instruction;  \* Rn: general purpose registers R0-R7 residing in one of the 4 register banks (located from 00h to 1FH in the 256-byte built-in RAM);  \* @Ri: indexing registers R0 and R1, in which the content of R0/R1 is treated as address of the targeted OPND, instead of the OPND itself;  \* bit8: where *bit8*is an 8-bit unsigned binary value interpreted as the address of every individual bit in Bit-Map area, or of certain registers in SPF-area (the accumulator, A, for instance; | | | |

**4. From Source to Machine Codes**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a typical 51-code example | | | | |
| source codes | | | machine codes | effects |
| label: | instructions | ; comments |
| ; TASK:  ; 1. shuffling 2 blocks of 8-byte data, respectively starting  ; at 30H and 40H;  ; 2. saving the shuffled 16-byte data in the block at 50H.  ; VARIABLEs in the code  ; R0: index for BLK30 retrieval  ; index for BLK50 deposit  ; R1: index for BLK40 retrieval  ; R2: counter for 8-rounded retrieval-deposit op.  ; 60H: temporal storage for BLK30 retrieval indexing  ; 61H: indexing for BLK50 deposit  ORG 0000H  mov SP, #70H .  mov R0, #30H  mov R1, #40H  mov 61H, #50H  mov R2, #8H  next\_byte:  mov A, @R0  inc R0  mov 60H, R0  mov R0, 61H  mov @R0, A ; BLK1 moving  inc 61H  mov A, @R1  inc R1  mov R0, 61H  mov @R0, A  inc 61H  mov R0, 61H  djnz R2, next\_byte  . . .  end | | | // code siz?  75H, 81H, 70H  78H, 30H  79H, 40H  75H, 61H, 50H  7AH, 08H  E6H  08H  88H, 60H  A8H, 61H  F6H  05H, 61H  E7H  09H  A8H, 61H  F6H  05H, 61H  A8H, 61H  DAH, ECH |  |