제06강

스택 및 서브루틴

스택 및 서브루틴

코드 변환

실습과제

ref.) Chapter 6

스택(stack)

: 임시 데이터의 저장이나 서브루틴의 호출에서 사용

: 스택영역은 내부 데이터 메모리에 위치

* SP(stack pointer)

  : 스택영역을 지시하는 8비트 레지스터

  : Reset시 내정된 SP의 값은 07H

  주의) 레지스터뱅크의 변경시, SP 값을 재설정 필요

```
MOV PSW, #00010000B      ; RB2 설정
MOV SP, #17H             ; 필히 재설정
```

스택(계속)

* PUSH와 POP 명령

: 1바이트 단위로 스택에 삽입, 혹은 인출

```
PUSH <직접주소>   ; SP를 +1후, SP위치에 기록.
POP <직접주소>    ; SP위치에서 인출후, SP를 -1함.
```

: 스택에 PUSH할 때 주소 증가방향으로 확장됨을 유의!!

: 범용레지스터(Rn)는 필히 직접주소로 표현

```
PUSH R0,        POP R0          ;사용불가
=> PUSH 00H,    POP 00H         ; 직접주소로
```

: A, B는 ACC, RB로 표현(어셈블러에 따라 다를 수 있음)

```
PUSH A,         POP A           ;사용불가
=> PUSH ACC,    POP ACC         ; 0AH와 혼동 피하도록
PUSH B,         POP B           ;사용불가
=> PUSH RB,     POP RB          ; 0BH와 혼동 피하도록
```

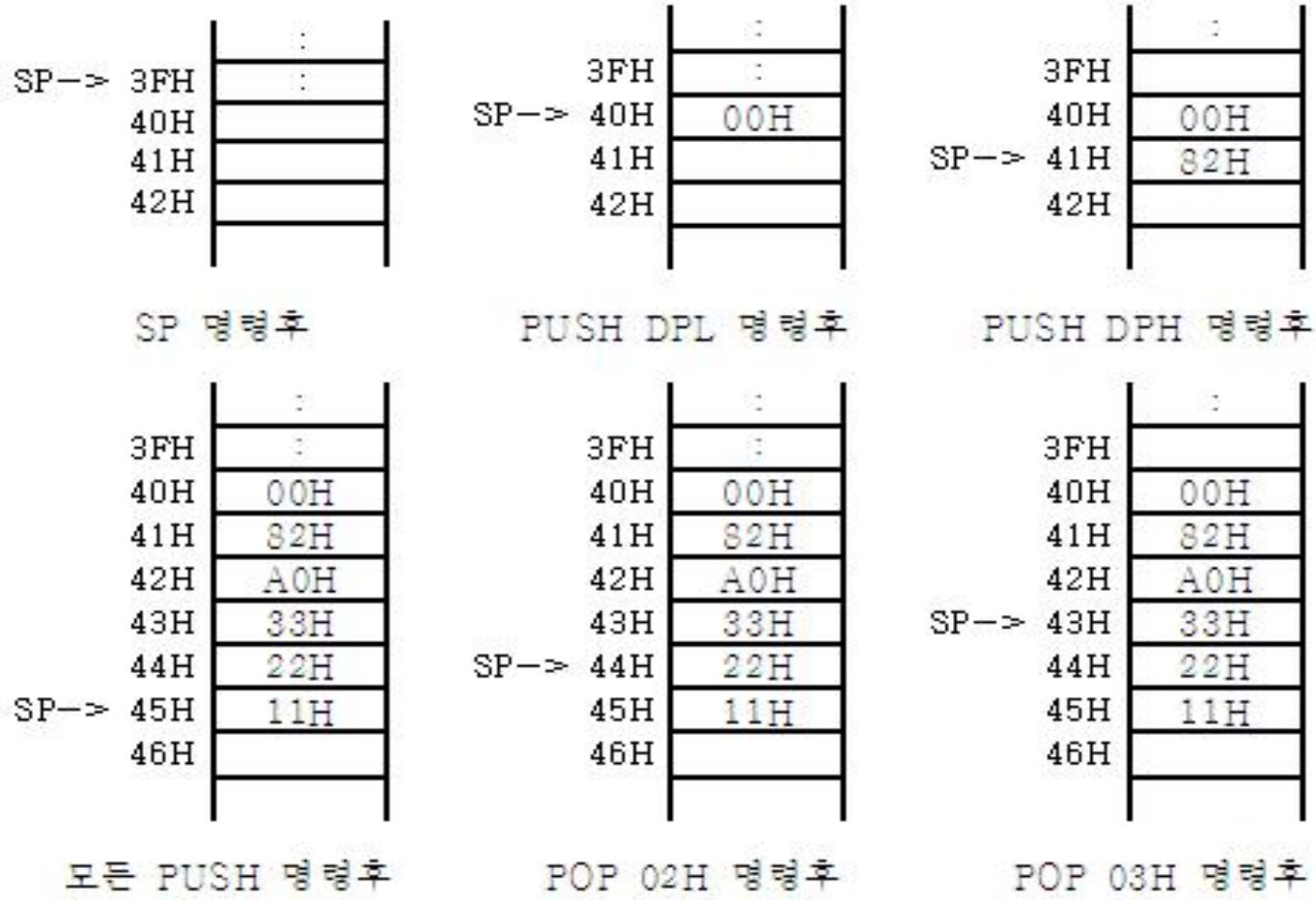
스택(계속)

* 스택의 동작

```
MOV     SP,#3FH          ; SP 설정
;
MOV     DPTR, #8200H
MOV     30H, #0A0H
MOV     A, #33H
MOV     R3, #22H
MOV     R2, #11H
;
PUSH    DPL              ; 스택에 데이터 저장
PUSH    DPH
PUSH    30H
PUSH    ACC              ; PUSH A 하지 말 것!
PUSH    03H
PUSH    02H
;
POP     02H              ; 스택의 데이터 인출
POP     03H
POP     ACC              ; POP A 하지 말 것!
POP     30H
POP     DPH
POP     DPL
;
```

스택(계속)

* 스택의 동작


SP-> 3FH
40H
41H
42H
SP 명령후
3FH
SP-> 40H
41H
42H
00H
PUSH DPL 명령후
3FH
40H
SP-> 41H
42H
00H
82H
PUSH DPH 명령후
3FH
40H
41H
42H
43H
44H
SP-> 45H
46H
00H
82H
A0H
33H
22H
11H
모든 PUSH 명령후
3FH
40H
41H
42H
43H
SP-> 44H
45H
46H
00H
82H
A0H
33H
22H
11H
POP 02H 명령후
3FH
40H
41H
42H
SP-> 43H
44H
45H
46H
00H
82H
A0H
33H
22H
11H
POP 03H 명령후

서브루틴

* 서브루틴의 정의

```
RTN:        ....
            ....
            RET
```

* 서브루틴 호출

```
LCALL RTN
ACALL RTN
    ; PC값의 상위 5비트+서브루틴주소의 하위 11비트 조합
CALL RTN
    ; PC값의 상위 8비트+서브루틴주소의 하위 8비트 조합
```

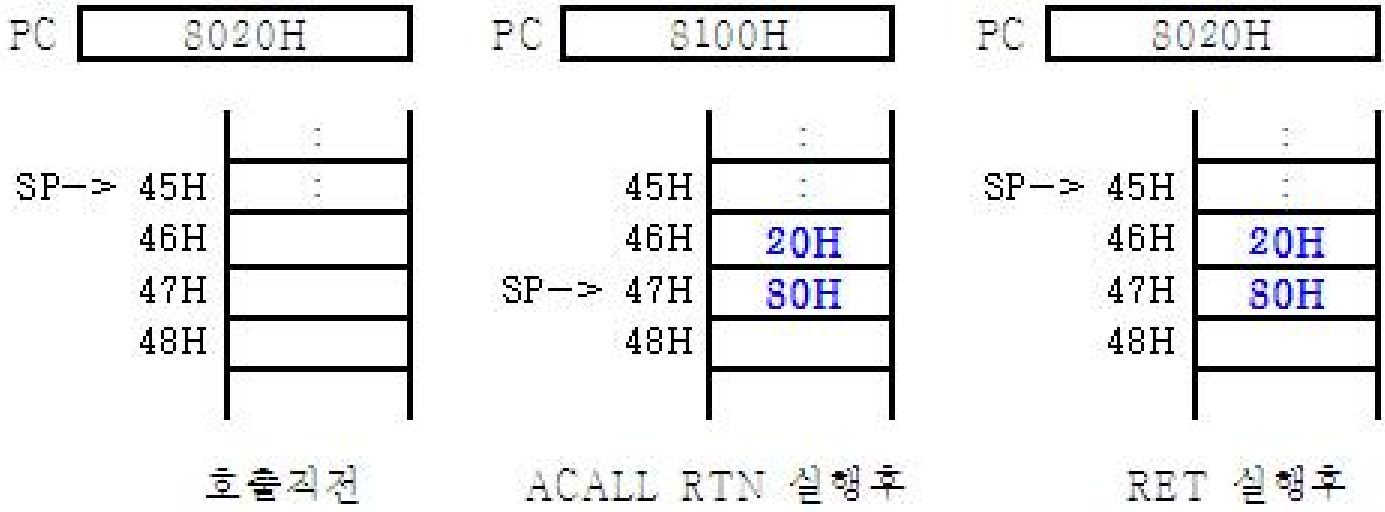
: 호출명령들은 PC의 하위바이트부터 스택에 저장

서브루틴(계속)

* 서브루틴 호출 및 반환 동작

: RTN 서브루틴을 호출할 때, PC의 값은 8020H이고, SP는 45H라 가정

```
        ⋮
        ACALL RTN        // call RTN
        ⋮
        ORG 8100H
RTN:    ⋮
        ⋮
        RET
```



서브루틴(계속)

* 메인루틴에서 사용되는 레지스터의 내용을 서브루틴 호출 후에도 유지하여야 하는 경우

```
        ....            ; A, B 사용중이라 가정
        CALL RTN
        ....            ; RTN 호출전의 A, B 사용해야..

RTN:    PUSH ACC        ; 스택에 퇴피
        PUSH RB         ; 스택에 퇴피
        ....            ; 퇴피된 A,B를 자유로이 사용
        ....
        ....
        ....
        POP RB          ; 원상복구
        POP ACC         ; 원상복구
        RET
```

코드변환

* 개요

[ASCII 코드 변환 테이블]

Char.	ASCII Code	비 고
'0'	30H	=val(Char)+30H
'1'	31H	
'2'	32H	
:	:	
'9'	39H	
'A'	41H	=val(Char)+30H+07H
'B'	42H	
:	:	
'F'	46H	

코드변환

1) 계산에 의한 방식

: 코드 변환 체계가 비교적 규칙적일 때 유용

; 규칙성을 파악하여 계산식 유도 필요

: 코드 테이블을 위한 메모리, 혹은 부가적인 하드웨어 불필요

: 반면, 연산과정에서 처리시간 상대적으로 더 요구

예) 코드변환 : 앞의 테이블 참조

; '0'~'9'와 'A'~'F'를 별도 처리

코드변환

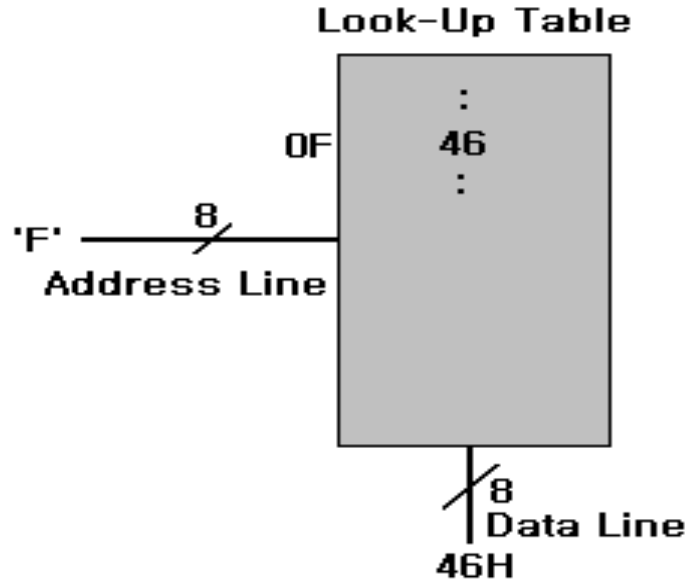
2) LUT(Look up table)에 의한 방식

: 코드 변환체계가 불규칙적일 때 유용

: 하드웨어 구현에 의한 고속 변환이 가능

: 코드 변환 테이블을 저장할 메모리를 추가 요구

예) 문자 'F'에 대한 ASCII코드 변환



예제실습

[예제1] 계산에 의한 코드 변환

: CHAR2HEX("F9" to Hex, p.174)

```
                          1     ;=========================================
                          2     ;   CC_01.ASM, Code Conversion
                          3     ;     CHAR2HEX by calculating
                          4     ;=========================================
                          5     $mod51
8000                      6             ORG 8000H
8000 758140               7             MOV SP,#40H
8003 74F9                 8             MOV A,#0F9H     ; 'F','9'
                          9
8005 F5F0                10             MOV B,A
8007 54F0                11             ANL A,#0F0H     ; masking...
8009 03                  12             RR A            ; SWAP A
800A 03                  13             RR A
800B 03                  14             RR A
800C 03                  15             RR A            ; 상위니블,'F'
800D 1123                16             CALL TOHEX
800F F8                  17             MOV R0,A
                         18
8010 E5F0                19             MOV A,B
8012 540F                20             ANL A,#0FH      ; 하위니블,'9'
```

```
8014 1123                21          CALL TOHEX
8016 F9                  22          MOV R1,A
                         23
8017 908100              24          MOV DPTR,#8100H       ;변환결과 기록
801A E8                  25          MOV A,R0
801B F0                  26          MOVX @DPTR,A
                         27
801C 0582                28          INC DPL
801E E9                  29          MOV A,R1
801F F0                  30          MOVX @DPTR,A
                         31
8020 020000              32          JMP 0000H
                         33
                         34          ; subroutine
8023 C3                  35    TOHEX: CLR C
8024 940A                36          SUBB A,#0AH
8026 4002                37          JC TO09           ; <10
8028 2407                38          ADD A,#07H
802A 2430                39    TO09: ADD A,#'0'        ; #30H
802C 240A                40          ADD A,#0AH
802E 22                  41          RET
                         42
                         43          END
```

예제실습(계속)

[예제2] 계산에 의한 코드 변환

: HEX2CHAR(44H, 37H to "7D", p.176)

```
                          1     ;=========================================
                          2     ;   CC_02.ASM, Code Conversion
                          3     ;     HEX2CHAR by calculating
                          4     ;=========================================
                          5     $mod51
8000                      6             ORG 8000H
8000 758140               7             MOV SP,#40H
                          8
8003 7844                 9             MOV R0,#44H     ; Hexa of 'D'
8005 7937                10             MOV R1,#37H     ; Hexa of '7'
                         11
8007 E8                  12             MOV A,R0
8008 111C                13             CALL TOCHAR
800A F5F0                14             MOV B,A
                         15
800C E9                  16             MOV A,R1
800D 111C                17             CALL TOCHAR
800F 03                  18             RR A             ; SWAP
8010 03                  19             RR A
8011 03                  20             RR A
```

```
8012 03                 21              RR A
8013 45F0               22              ORL A,B
                        23
8015 908100             24              MOV DPTR,#8100H
8018 F0                 25              MOVX @DPTR,A
                        26
8019 020000             27              JMP 0000H
                        28
                        29              ; subroutine
801C C3                 30      TOCHAR: CLR C
801D 9430               31              SUBB A,#'0'     ; -30H
801F FF                 32              MOV R7,A
8020 C3                 33              CLR C
8021 940A               34              SUBB A,#0AH     ; 비교
8023 4005               35              JC TO09
8025 EF                 36              MOV A,R7
8026 C3                 37              CLR C
8027 9407               38              SUBB A,#07H     ; -7H
8029 22                 39              RET
802A EF                 40      TO09:   MOV A,R7
802B 22                 41              RET
                        42
                        43              END
```

예제실습(계속)

[예제3] LUT에 의한 코드 변환

: CHAR2HEX(p.177)

```
                          1     ;=========================================
                          2     ;   CC_03.ASM, Code Conversion
                          3     ;     CHAR2HEX by LUT
                          4     ;=========================================
                          5     $mod51
8000                      6             ORG 8000H
8000 758140               7             MOV SP,#40H
8003 74A7                 8             MOV A,#0A7H     ; 'A','7'
                          9
8005 F5F0                10             MOV B,A
8007 54F0                11             ANL A,#0F0H
8009 03                  12             RR A            ; SWAP A
800A 03                  13             RR A
800B 03                  14             RR A
800C 03                  15             RR A
800D 1123                16             CALL TOHEX
800F F8                  17             MOV R0,A
                         18
8010 E5F0                19             MOV A,B
8012 540F                20             ANL A,#0FH
```

```
8014 1123          21              CALL TOHEX
8016 F9            22              MOV R1,A
                   23
8017 908100        24              MOV DPTR,#8100H
801A E8            25              MOV A,R0
801B F0            26              MOVX @DPTR,A
                   27
801C 0582          28              INC DPL
801E E9            29              MOV A,R1
801F F0            30              MOVX @DPTR,A
                   31
8020 020000        32              JMP 0000H
                   33
                   34              ; subroutine
8023 908050        35      TOHEX:  MOV DPTR,#TABLE      ; 베이스주소
8026 93            36              MOVC A,@A+DPTR
8027 22            37              RET
                   38
8050               39              ORG 8050H
8050 30313233      40      TABLE:  DB 30H,31H,32H,33H
8054 34353637      41              DB 34H,35H,36H,37H
8058 38394142      42              DB 38H,39H,41H,42H
805C 43444546      43              DB 43H,44H,45H,46H
                   44              END
```

예제실습(계속)

[예제4] LUT에 의한 코드 변환

: HEX2CHAR(p.179)

```
                          1     ;=========================================
                          2     ;   CC_04.ASM, Code Conversion
                          3     ;     HEX2CHAR by LUT
                          4     ;=========================================
                          5     $mod51
8000                      6             ORG 8000H
8000 75812F               7     BASE:   MOV SP,#2FH
                          8
8003 7844                 9             MOV R0,#44H     ; Hexa of 'D'
8005 7937                10             MOV R1,#37H     ; Hexa of '7'
                         11
8007 E8                  12             MOV A,R0
8008 1119                13             CALL TOCHAR
800A F5F0                14             MOV B,A
                         15
800C E9                  16             MOV A,R1
800D 1119                17             CALL TOCHAR
                         18
800F C4                  19             SWAP A
                         20
```

```
8010 45F0          21              ORL A,B
8012 908100        22              MOV DPTR,#8100H
8015 F0            23              MOVX @DPTR,A
                   24
8016 020000        25              JMP 0000H
                   26
                   27              ; subroutine
8019 908000        28      TOCHAR: MOV DPTR,#BASE          ; 베이스주소
801C 2582          29              ADD A,DPL
801E F582          30              MOV DPL,A
8020 E0            31              MOVX A,@DPTR
8021 22            32              RET
                   33
8030               34              ORG 8030H
8030 00010203      35      TABLE:  DB 00H,01H,02H,03H
8034 04050607      36              DB 04H,05H,06H,07H
8038 0809          37              DB 08H,09H
                   38
8041               39              ORG 8041H
8041 0A0B          40      TABLE1: DB 0AH,0BH
8043 0C0D0E0F      41              DB 0CH,0DH,0EH,0FH
                   42
                   43              END
```

보완) LUT의 기준위치를 8200H로 변경하여..

실습과제

[실습1] 스택활용한 역순 재배열

[과제] 블록데이터 전송

: 외부 데이터 메모리 영역 -> 내부 램 영역 ->
또 다른 외부 데이터 메모리 영역으로 블록 전송

: 다음 부분을 서브루틴으로 각각 구성

1) 외부 메모리 영역 -> 내부 메모리 영역 전송

2) 내부 메모리 영역 -> 또 다른 외부 메모리 영역

: 기타 사항은 각자 정의

실습과제(계속)

[실습2] Char2Hex 코드 변환

: "F1FA2002"을 계산에 의한 방식으로

[실습3] Char2Hex 코드 변환

: "F1FA2002"을 LUT에 의한 방식으로

실습과제(계속)

[실습4] Hex2Char 코드 변환

: [실습2] 혹은 [실습3]의 결과에 대해

: 계산에 의한 방식으로

[실습5] Hex2Char 코드 변환

: [실습2] 혹은 [실습3]의 결과에 대해

: LUT에 의한 방식으로

실습과제(계속)

[실습6] 코드변환

: 2개 문자로 구성된 1바이트가 외부메모리 공간에 위치함.

: 이에 대한 ASCII 코드를 구하여 외부메모리에 저장.

: 저장된 ASCII 코드 2바이트를 역으로 변환하여
1바이트로 구성된 결과를 또 다른 외부메모리에 저장

: 처리방법(계산 혹은 LUT)은 자유선택

: 제반 필요한 사항들은 각자 정의할 것