실험제목 : ATmega328PB 어셈블리 언어 실습 (II)

실험목적

ATmega328PB의 명령어의 동작을 이해하고 이들을 조합하여 간단한 프로그램을 작성하는 실습 과정을 통해 ATmega328PB의 내부 구조를 이해하고 어셈블리 언어로 프로그래밍하는 능력을 배양한다.

실험 준비물

Atmel Studio 7

실험에 필요한 예비지식

1. 실험에 필요한 자료
   1) 강의 노트 파일 참조
   2) AVR-Instruction-Set-Manual
   3) AVR Assembler User Guide (AVRASM2)

2. Assembly Language Program Format (AVRASM2)

1) 형식

```
[label:] directive [operands] [Comment]
[label:] instruction [operands] [Comment]
Comment
Empty line
```

2) 설명

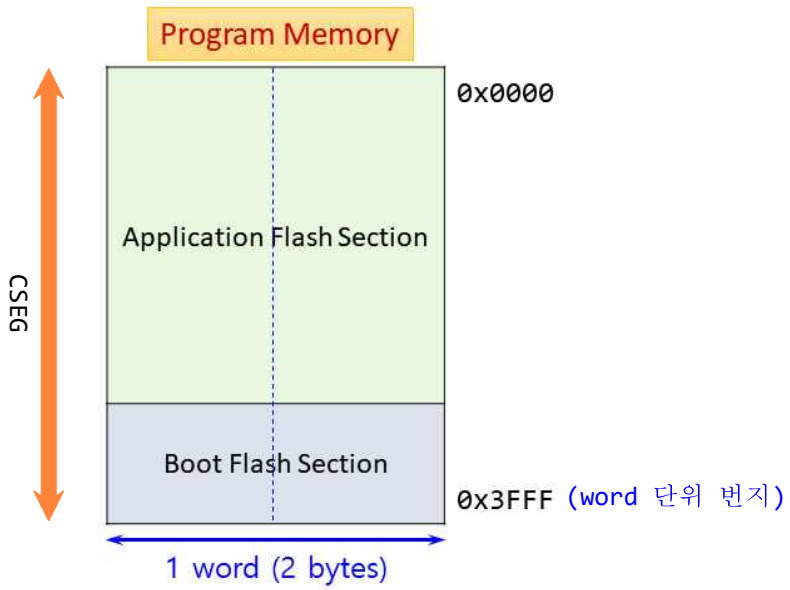
가) [ ] 부분은 생략 가능.

나) Label은 반드시 colon(:)으로 끝나야 함.

나) Comment는 반드시 semicolon(;)으로 시작해야 함.

3. Assembler Directives

(참고) ATmega328PB Memory Map



1) CSEG

가) 용도

Code segment의 시작을 알림.

Code segment만의 location counter를 가지며 byte counter가 아닌 word counter이다.

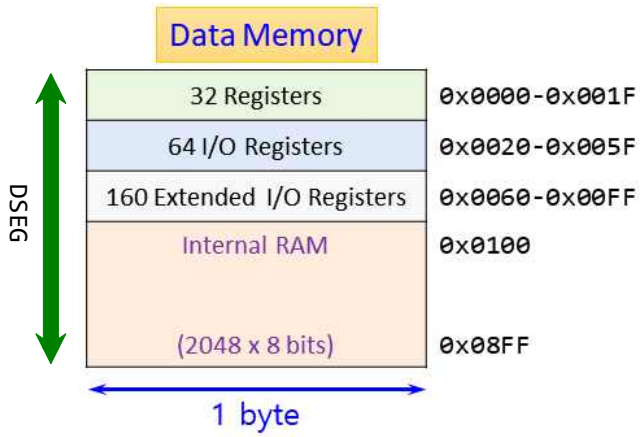
나) 형식

```
.CSEG
```

다) 예

```
            .DSEG
            .ORG        0x200   ; byte 단위 번지
BUF:        .BYTE       4
            .CSEG
            .ORG        0x100   ; word 단위 번지
START:      LDI         R15,0x6F
             ...
```

2) DSEG



가) 용도

Data segment의 시작을 알림

Data segment만의 location counter를 가지며 byte counter이다.

나) 형식

```
.DSEG
```

다) 예

```
            .DSEG
            .ORG        0x200
BUF:        .BYTE       4
            .CSEG
            .ORG        0x100
START:      LDI         R15,0x6F
            ...
```

3) ORG

가) 용도

Location Counter를 지정된 값으로 초기화

나) 형식

```
.ORG
```

다) 예

```
            .DSEG
            .ORG        0x200   ; byte 단위 번지
BUF:        .BYTE       4
            .CSEG
            .ORG        0x100   ; word 단위 번지
START:      LDI         R15,0x6F
             ...
```

4) BYTE

가) 용도

임시 데이터 저장용 공간을 확보하기 위해 Data Segment (SRAM) 영역에 지정된 바이트 수 만큼 메모리 공간을 확보

나) 형식

```
.BYTE   n
```

다) 예

```
            .DSEG
            .ORG        0x200
BUF:        .BYTE       4
            .CSEG
            .ORG        0x100
START:      LDI         R15,0x6F
             ...
```

5) DB, DW, DD, DQ

가) 용도

상수값을 저장하기 위해 Program Memory나 EEPROM 영역에 지정된 바이트(혹은 워드) 수 만큼 메모리 공간을 확보

나) 형식

```
.DB     n
.DW     n
```

다) 예

```
            .DSEG
            .ORG        0x200
BUF:        .BYTE       4
            .CSEG
            .ORG        0x100
LED_TBL:    .DB         0x7F, 48, 0b01010101, -128
WD_TBL:     .DW         0x1234, 32764
START:      LDI         R15,0x6F
             ...
```

6) DEF

가) 용도

변용 레지스터에 다른 이름(symbolic name)을 부여할 때 사용.

나) 형식

```
.DEF  Symbol = Register
```

다) 예

```
        .DEF    TEMP=R16
        .DEF    IOR=R0
        LDI     TEMP,0xF0       ; same as LDI R16,0xF0
        IN      IOR,0x3F        ; Read SREG into IOR register
        EOR     TEMP,IOI        ;
             ...
```

7) EQU

가) 용도

일반적으로 상수값을 정의하는데 사용

나) 형식

```
.EQU  Symbol = expression
```

다) 예

```
        .EQU    LED=5
        .EQU    DDRB=0x04
        .EQU    PORTB=0x05
        SBI     DDRB,LED        ; same as SBI DDRB,5
        SBI     PORTB,LED       ; Set PB5 to HIGH
             ...
```

4. 실험에 필요한 명령어

1) Load Immediate

가) 형식

```
LDI Rd,K
```

나) 동작

Rd ← K 단, 16≤d≤31, 0≤K≤255

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

라) 예

```
LDI R16,0xF5
```

2) Add without Carry

가) 형식

```
ADD Rd,Rr
```

나) 동작

Rd ← Rd + Rr 단, 0≤d≤31, 0≤r≤31

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	↔	↔	↔	↔	↔	↔

라) 예

```
ADD R16,R17
```

3) Subtract without Carry

가) 형식

```
SUB Rd,Rr
```

나) 동작

Rd ← Rd - Rr 단, 0≤d≤31, 0≤r≤31

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	↔	↔	↔	↔	↔	↔

라) 예

```
SUB R16,R17
```

4) Multiply Unsigned

가) 형식

```
MUL Rd,Rr
```

나) 동작

R1:R0 ← Rd × Rr 단, 0≤d≤31, 0≤r≤31

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	↔	↔

라) 예

```
MUL R16,R17
```

5) Store Direct to Data Space

가) 형식

```
STS addr16,Rr
```

나) 동작

(addr16) ← Rr 단, 0≤r≤31, 0≤addr16≤65535

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

라) 예

```
STS 0x0102,R17
```

6) Relative Jump

가) 형식

```
RJMP k
```

나) 동작

PC ← PC + k + 1 단, -2,048≤k≤+2,047

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

라) 예

```
HERE:  RJMP HERE
```

7) Add with Carry

가) 형식

```
ADC Rd,Rr
```

나) 동작

Rd ← Rd + Rr + C 단, 0≤d≤31, 0≤r≤31

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	↔	↔	↔	↔	↔	↔

라) 예

```
ADC R16,R17
```

8) Subtract with Carry

가) 형식

```
SBC Rd,Rr
```

나) 동작

Rd ← Rd - Rr - C 단, 0≤d≤31, 0≤r≤31

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	↔	↔	↔	↔	↔	↔

라) 예

```
SBC R16,R17
```

9) Load Program Memory

가) 형식 및 동작

LPM	R0 ← (Z)	
LPM Rd,Z	Rd ← (Z)	단, 0≤d≤31
LPM Rd,Z+	Rd ← (Z), Z ← Z + 1	단, 0≤d≤31

나) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

다) 예

```
        LDI     ZH,HIGH(TABLE << 1)   ; Initialize Z-pointer, R31
        LDI     ZL,LOW(TABLE << 1)    ; R30
        LPM     R16,Z+
HERE:   RJMP    HERE
        .ORG    0x100
TABLE:  .DB     0x13, 0x57, 0x9B, 0xDF
```

10) Store Indirect From Register to Data Space using Index X

가) 형식 및 동작

ST X,Rr	(X) ← Rr	단, 0≤r≤31
ST X+,Rr	(X) ← Rr, X ← X + 1	단, 0≤r≤31
ST -X,Rr	X ← X - 1, (X) ← Rr,	단, 0≤r≤31

나) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

다) 예

```
        .CSEG
        LDI     XH,HIGH(ANS_4)          ; Initialize X-pointer, R27
        LDI     XL,LOW(ANS_4)           ; R26
        ST      X+,R16                  ; Store data to SRAM pointed by X
HERE:   RJMP    HERE
        .DSEG
        .ORG    0x0100
ANS_4:  .BYTE   4
```

11) Branch if Not Equal

가) 형식

```
BRNE k                  단, -64≤k≤+63
```

나) 동작

```
PC ← PC + k + 1         if condition is true
PC ← PC + 1             if condition is false
```

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

라) 예

```
        LDI     R17,5
EX4_1:  DEC     R17
        ...
        BRNE    EX4_1           ; Repeat 5 times
```

12) Clear Register

가) 형식

```
CLR Rd
```

나) 동작

Rd ← Rd XOR Rd 단, 0≤d≤31

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	0	0	0	1	-

라) 예

```
CLR R16
```

13) Decrement

가) 형식

```
DEC Rd
```

나) 동작

Rd ← Rd - 1 단, 0≤d≤31

나) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	↔	↔	↔	↔	-

라) 예

```
DEC R16
```

14) Increment

가) 형식

```
INC Rd
```

나) 동작

Rd ← Rd + 1 단, 0≤d≤31

나) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	↔	↔	↔	↔	-

라) 예

```
INC R16
```

15) Two's Complement

가) 형식

```
NEG Rd
```

나) 동작

Rd ← 0x00 - Rd 단, 0≤d≤31

나) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	↔	↔	↔	↔	↔	↔

라) 예

```
NEG R16
```

16) Copy Register

가) 형식

```
MOV Rd,Rr
```

나) 동작

Rd ← Rr 단, 16≤d≤31, 16≤r≤31

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

라) 예

```
LDI R16,R15
```

17) Bitwise OR

가) 형식

```
OR Rd,Rr
```

나) 동작

Rd ← Rd OR Rr 단, 0≤d≤31, 0≤r≤31

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	↔	0	↔	↔	-

라) 예

```
OR R16,R17
```

18) Bitwise AND

가) 형식

```
AND Rd,Rr
```

나) 동작

Rd ← Rd AND Rr 단, 0≤d≤31, 0≤r≤31

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	↔	0	↔	↔	-

라) 예

```
AND R16,R17
```

19) Bitwise AND with Immediate

가) 형식

```
ANDI Rd,K
```

나) 동작

Rd ← Rd AND K 단, 16≤d≤31, 0≤K≤255

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	↔	0	↔	↔	-

라) 예

```
ANDI R16,0x0F
```

20) Set Bit in I/O Register

가) 형식

```
SBI A,b
```

나) 동작

```
I/O(A,b) ← 1          단, 0≤A≤31,    0≤b≤7
```

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

라) 예

```
SBI PORTB,5    ; Set PB5
```

21) Clear Bit in I/O Register

가) 형식

```
CBI A,b
```

나) 동작

```
I/O(A,b) ← 0          단, 0≤A≤31,    0≤b≤7
```

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

라) 예

```
CBI PORTB,5    ; Clear PB5
```

22) Relative Call to Subroutine

가) 형식

RCALL addr2k 단, -2,048≤addr2k≤+2,047

나) 동작

(SP) ← PC + 1, SP ← SP - 2, PC ← PC + addr2k + 1,

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

라) 예

```
            RCALL   DELAY_MS
HERE:       RJMP    HERE
;
DELAY_MS:   ...
            RET
```

23) Return from Subroutine

가) 형식

RET

나) 동작

PC ← (SP), SP ← SP +2

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

라) 예

위 RCALL 명령어 예제 참조

24) Push Register on Stack

가) 형식

PUSH Rr 단, 0≤r≤31

나) 동작

(SP) ← Rr, SP ← SP-1, PC ← PC + 1

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

라) 예

```
PUSH    R23
...
POP     R23
```

25) Pop Register from Stack

가) 형식

POP Rd 단, 0≤d≤31

나) 동작

Rd ← (SP), SP ← SP+1, PC ← PC + 1

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

라) 예

위 PUSH 명령어 예제 참조

26) Subtract Immediate from Word

가) 형식

```
SBIW Rd+1,Rd,K
```

나) 동작

R[d+1]:Rd ← R[d+1]:Rd - K d∈{24, 26, 28, 30}, 0≤K≤63

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	↔	↔	↔	↔	↔

라) 예

```
SBIW R25:R24,1
SBIW R24,1          ; 위의 명령어와 같은 의미
```

27) Store Register to I/O Location

가) 형식

```
OUT A,Rr
```

나) 동작

I/O(A) ← Rr 단, 0≤r≤31, 0≤A≤63

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

라) 예

```
OUT SPH,R16    ; Stack Pointer (SPH)
```

28) No Operation

가) 형식

```
NOP
```

나) 동작

없음

다) Status Register (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

라) 예

```
NOP
```

5. 상태 레지스터(Status Register, SREG)의 각 비트의 의미

Bit No.	7	6	5	4	3	2	1	0
Name	I	T	H	S	V	N	Z	C
Reset	0	0	0	0	0	0	0	0

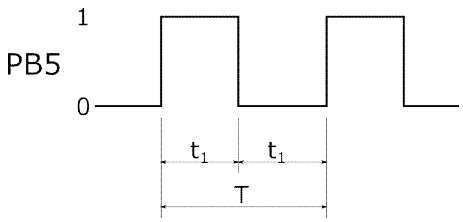
Bit 7 - I: Global Interrupt Enable
Bit 6 - T: Copy Storage
Bit 5 - H: Half Carry Flag
Bit 4 - S: Sign Flag. S=N⊕V.
Can be used to determine if the result of a previous operation is positive or negative.
Bit 3 - V: Two's Complement Overflow Flag
Bit 2 - N: Negative Flag. Same as bit 7 of previous operation.
Bit 1 - Z: Zero Flag
Bit 0 - C: Carry Flag

실습 내용

1. 조건부 분기 명령어를 사용하여 F_CPU가 1 MHz일 때 약 1 msec의 시간 지연이 되도록 어셈블리 언어로 프로그램을 작성하시오. (LDI, SBIW, BRNE, RJMP 명령어 사용)

2. 다음 프로그램을 실행시키면 PB5에 아래 그림과 같은 펄스 신호가 무한히 반복해서 출력된다.

```
        SBI     DDRB,5    ; set PB5 to OUTPUT mode
LP1:    SBI     PORTB,5   ; set PB5 to HIGH
        NOP
        CBI     PORTB,5   ; set PB5 to LOW
        NOP
        RJMP    LP1
```



(1) ATmega328PB instruction manual을 참고하여 출력 파형이 논리값 '1'로 유지되는 시간과 논리값 '0'으로 유지되는 시간 및 주기를 계산하여 아래의 표에 기록하시오.

구간	클럭 수
t_1	
t_2	
T	

(2) ATmega328PB Xmini board에서 이 프로그램을 실행시킨 후, oscilloscope를 사용하여 출력 파형이 논리값 '1'로 유지되는 시간과 논리값 '0'으로 유지되는 시간 및 주기를 관찰하여 아래의 표에 기록하고 위 (1)의 계산값과 비교하시오. 단, ATmega328PB의 주파수를 16 MHz로 설정하시오.

구간	시간 (nsec)
t_1	
t_2	
T	

3. (Subroutine 작성 및 이를 이용한 LED ON/OFF 제어) 위 1에서 작성한 1 msec 시간 지연 프로그램을 delay_1ms라는 이름의 subroutine으로 만들고 이를 이용하여 ATmega328PB board의 PB5에 연결된 LED가 50 msec 동안 점등되고, 950 msec 동안 소등되는 동작을 무한히 반복하는 프로그램을 어셈블리 언어로 프로그램을 작성하시오.

(1) delay_1ms subroutine 내부에서 사용하는 Register들을 Stack Memory에 PUSH/POP 명령어를 사용하여 원래의 값을 보존.

(2) ATmega328PB의 주파수를 16 MHz로 설정.

```
; (1) 필요한 초기화 작업

; (2) LED 점등

; (3) 50 msec 동안 시간 지연
;     delay_1ms subroutine을 50회 반복하여 호출

; (4) LED 소등

; (5) 950 msec 동안 시간 지연
;     delay_1ms subroutine을 950회 반복하여 호출

; (6) 위 (2)-(5)의 동작을 무한히 반복

;-----------------------------------
; 1 msec 시간 지연 함수
delay_1ms:
; PUSH 명령어를 사용하여 이 subroutine에서 사용하는 모든 레지스터의 값을
; 스택 메모리에 저장



; POP 명령어를 사용하여 스택 메모리에 저장되었던 모든 레지스터의 값을 복원

                ret
;-----------------------------------
```

4. 위 6의 프로그램에서는 stack memory를 사용하게 된다. (1) delay_1ms subroutine을 호출하기 직전과 (2) delay_1ms subroutine 처음 부분 (3) delay_1ms subroutine에서 돌아온 직후의 SP(stack pointer)의 값과 내부 데이터 메모리 0x8FE와 0x8FF 번지의 내용을 아래의 표에 기록하시오.

	SP 값	내부 데이터 메모리 0x8FE 번지의 내용	내부 데이터 메모리 0x8FF 번지의 내용
(1) delay_1ms subroutine을 호출하기 직전			
(2) delay_1ms subroutine 처음 부분			
(3) delay_1ms ubroutine에서 돌아온 직후			

과제 (보고서)

실습 내용 1, 2, 3, 및 4의 실습 내용과 결과를 정리하여 보고서로 제출

-끝-