어셈블리 Take-home Exam

문항 수: 10 2020년 2학기

0[배점 없음]. 학번 및 이름을 쓰시오.

학번: **B711222** 이름: **박조은**

1[2]. SPARC 프로세서는 다음 중 어느 기계에 속하는가?

① : 스택 기계 (stack machine)
② : 단일 레지스터 기계 (accumulator machine)
③ : 다중 레지스터 기계 (load/store machine)

2[8]. 다음 SPARC에 대한 설명의 참 / 거짓을 선택하시오.

① 전방전달(forwarding)을 이용하여도 파이프라인 지연을 막을 수 없는 경우가 있다. (참 / 거짓)

② 레지스터의 개수가 늘어나면, 기존에 구현할 수 없었던 프로그램 구현이 가능해지기도 한다. (참 / 거짓)

③ 메모리 장치와 프로세서 사이 가능한 데이터 이동단위는 1바이트, 2바이트, 그리고 4바이트이다. (참 / 거짓)

④ 레지스터 %l0 값이 13일 경우, 명령어 "btst 0x19, %l0"를 실행하면 condition code 중 Z값이 0으로 세팅된다. (참 / 거짓)

3[20]. 다음 SPARC 어셈블리 표현 중 잘못된 형식 여부를 참 / 거짓으로 표시하고, 거짓인 경우는 잘못된 부분을 설명하시오.

① add 10, %l0, %l1 (참 / 거짓)

원래 순서: add regrs1, reg_or_imm, regrd
오답 수정: add %l0, 10, %l1

② addcc %l0, %l1, %l2 (참 / 거짓)

③ ld %l0, [%l1 + 1] (참 / 거짓)

원래 순서: ld [address], regrd
오답 수정: ld [%l1 + 1], %l0

④ st [%l0 + 0], %o0 (참 / 거짓)

원래 순서: st regrd, [address]
오답 수정: st %o0, [%l0 + 0]

⑤ cmp %l0, 0x2345 (참 / 거짓)

⑥ VAL = 10 + 2 * 5 (참 / 거짓)

SPARC assembly level에서는 덧셈을 표현하기 위해 3개의 operand가 필요하며, 곱셈은 subroutine과 nop를 사용해야 한다. 또한 그 연산 값을 변수에 할당하기 위해서는 2개의 operand가 필요하다. 즉 ⑥번 식과 같은 간단한 연산도 여러 줄의 logic을 취할 수밖에 없다.

⑦ ld [%i1 – %l2], %l0 (참 / 거짓)

⑧ st %i5, [%l1 – 100] (참 / 거짓)

⑨ ldd [%o1+0x8], %o3 (참 / 거짓)

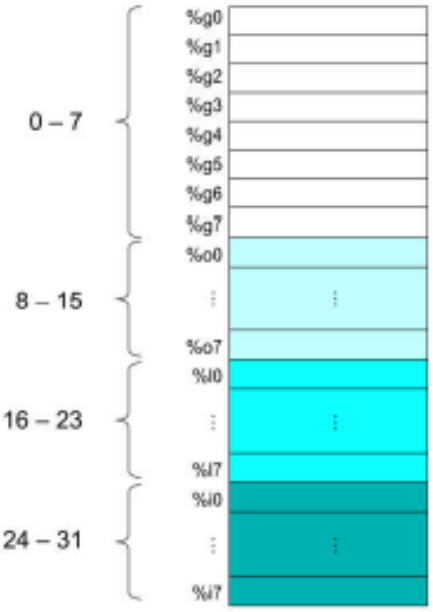
⑩ or %o2, %e2, %o3 (참 / 거짓)

ld 명령어의 형식은 다음과 같아야 하기 때문이다.
ld [R + A], S

%r0부터 %r31까지의 32개 register들은 순서대로 %g0~%g7, %o0~%o7, %l0~%l7, %i0~%i7의 다른 이름으로 표현 가능하다. 즉 e로 시작하는 register 이름은 없다.

4[10]. 다음 정보를 참고하시오.

※SPARC 레지스터 번호



※산술 및 논리 연산 명령어 형식

비트번호	31 30	29 25	24 19	18 14	13	12 5	4 0
의 미	OP	S	OP-확장	R	0		A

비트번호	31 30	29 25	24 19	18 14	13	12 0
의 미	OP	S	OP-확장	R	1	상수 (A)

OP 는 "10_2"

OP-확장:

명령어	OP-확장
add	000000
and	000001
or	000010
xor	000011
sub	000100
andn	000101
orn	000110
xnor	000111
addx	001000
subx	001100
addcc	010000
andcc	010001
orcc	010010

명령어	OP-확장
xorcc	010011
subcc	010100
andncc	010101
orncc	010110
xnorcc	010111
addxcc	011000
subxcc	011100
sll	100101
srl	100110
sra	100111
jmpl	111000
save	111100
restore	111101

4.1[5]. 다음 SPARC 어셈블리 명령어를 16진수(hex) 기계코드(machine code)로 표현하시오.

xnorcc %l1, -2, %l2

0xA4BC7FFE

4.2[5]. 다음 16진수(hex) 기계코드(machine code)를 SPARC 어셈블리 명령어로 표현하시오.

0x80A7C00F

subcc %i7, %o7, %g0

5[10]. 다음은 %l0 레지스터 값, %l1 레지스터의 값, 그리고 실행할 명령어이다. 명령어 실행 후, condition code Z, N, V, C 및 %l2 레지스터 값을 표에 적으시오.

5.1[5]. %l0: 0x8000_0000
%l1: 0x8000_0000

addcc %l0, %l1, %l2

Z	N	V	C	%l2
1	0	1	1	0x0000_0000

5.2[5]. %l0: 0x0123_4567
%l1: 0xFEDC_BA98

xorcc %l0, %l1, %l2

Z	N	V	C	%l2
0	1	0	0	0xFFFF_FFFF

6[10]. 다음 코드의 지연주기를 최적화하시오.
※ 힌트: 의사명령어 set을 두 개의 기계명령어로 분리한 후, 최적화하시오.

```
.text
.global main

main:
    save    %sp, -96, %sp
    set     str, %o0
    call    printf
    nop
    ret
    restore

.data
str: .asciz "Hello!\n"
.align 4
```

!! 답안을 여기에 적으시오.

```
1  .text
2  .global main
3
4  main:   save %sp, -96, %sp
5          sethi %hi(str), %o0
6          call printf
7          or %o0, %lo(str), %o0
8
9          ret
10         restore
11
12 .data
13 str: .asciz "Hello!\n"
14 .align 4
```

7[10]. 두 개의 양의 정수를 십진수 형태로 입력받아 두 수 중에서 작은 수부터 1씩 증가하여 큰 수까지 더하는 어셈블리 프로그램을 작성하고, 코드(mid_prob7.s)를 submit하시오. (명령: submit konwoo mid_07)

※ 가정: 사용자 입력은 32비트 부호 있는 정수 표현들 중 임의의 양수이다.

※ 동작 예시:

```
bash $ ./mid_prob7
Value?> 7
Value?> 3
Sum is 25
```

```
bash $ ./mid_prob7
Value?> 3
Value?> 7
Sum is 25
```

```
bash $ ./mid_prob7
Value?> 11
Value?> 20
Sum is 155
```

!! 답안을 여기에 적으시오. 공간이 부족할 경우, 2열(2-column)로 작성 또는 추가페이지 사용 가능.

```
.section ".text"
str1:   .asciz "Value?> "
str2:   .asciz "%d"
str3:   .asciz "Sum is %d\n"
.align  4
.global main, scanf, printf

main:   save %sp, -96, %sp
        mov %g0, %i0        ! int temp = 0
        mov %g0, %i1        ! int sum = 0

input:  set str1, %o0       ! print "Value?> " for a
        call printf
        nop

        set str2, %o0       ! input a
        add %fp, -4, %o1
        call scanf
        nop

        ld [%fp-4], %l0     ! return 0 if a <= 0
        subcc %l0, 0, %g0
        ble exit
        nop

        set str1, %o0       ! print "Value?> " for b
        call printf
        nop

        set str2, %o0       ! input b
        add %fp, -8, %o1
        call scanf
        nop

        ld [%fp-8], %l1     ! return 0 if b <= 0
        subcc %l1, 0, %g0
        ble exit
        nop

        subcc %l0, %l1, %g0 ! if a > b
        bg swap             ! jump to "swap" label
        nop

        subcc %l0, %l1, %g0 ! if a <= b
        ble loop            ! jump to "loop" label
        nop

swap:   mov %l1, %i0        ! b -> temp (temp = b)
        mov %l0, %l1        ! a -> b (b = a)
        mov %i0, %l0        ! temp -> a (a = temp)

loop:   subcc %l0, %l1, %g0 ! if a > b
        bg result           ! 루프 이탈
        nop

        add %i1, %l0, %i1   ! sum += a
        inc %l0             ! a++
        ba loop             ! 후방 분기
        nop

result: set str3, %o0       ! printing the result
        mov %i1, %o1
        call printf
        nop

exit:   ret
        restore
```

8[10]. (구현) 네 개의 32비트 정수(A, B, C, D)를 16진수 형태로 입력받아 A*B + C*D를 계산하는 프로그램을 작성하고, 코드(mid_prob8.s)를 submit하시오. (명령: submit konwoo mid_08)

※ 가정: 사용자 입력은 32비트 0 또는 양수이다. 결과는 64비트로 표현.

※ 동작 예시:

```
bash $ ./mid_prob8
Hexadecimal value?> ffffffff
Hexadecimal value?> 8
Hexadecimal value?> ffffffff
Hexadecimal value?> 8
Result is 0000000f fffffff0
```

```
bash $ ./mid_prob8
Hexadecimal value?> ffffffff
Hexadecimal value?> ffffffff
Hexadecimal value?> ffffffff
Hexadecimal value?> ffffffff
Result is fffffffc 00000002
```

!! 답안을 여기에 적으시오. 공간이 부족할 경우, 2열(2-column)로 작성 또는 추가페이지 사용 가능.

```
.section ".text"
str1:   .asciz "Hexadecimal value?> "
str2:   .asciz "%x"
str3:   .asciz "Result is %.8x %.8x"
.align  4
.global main, scanf, printf

main:   save %sp, -96, %sp

        set str1, %o0
        call printf
        nop

        set str2, %o0
        add %fp, -4, %o1
        call scanf          ! input 1st value
        nop
        ld [%fp-4], %l0

        set str1, %o0
        call printf
        nop

        set str2, %o0
        add %fp, -8, %o1
        call scanf          ! input 2nd value
        nop
        ld [%fp-8], %l1

        mov %l0, %o0
        mov %l1, %o1
        call .umul          ! multiplication
        nop
        mov %o0, %i0        ! $i0 = (1st val * 2nd val) 하위비트
        mov %o1, %i1        ! $i1 = (1st val * 2nd val) 상위비트

        set str1, %o0
        call printf
        nop

        set str2, %o0
        add %fp, -12, %o1
        call scanf          ! input 3rd value
        nop
        ld [%fp-12], %l2

        set str1, %o0
        call printf
        nop

        set str2, %o0
        add %fp, -16, %o1
        call scanf          ! input 4th value
        nop
        ld [%fp-16], %l3

        mov %l2, %o0
        mov %l3, %o1
        call .umul          ! multiplication
        nop
        mov %o0, %i2        ! $i2 = (3rd val * 4th val) 하위비트
        mov %o1, %i3        ! $i3 = (3rd val * 4th val) 상위비트

        addcc %i0, %i2, %i0 ! $i0 = 하위비트끼리 더한 값
        addx %i1, %i3, %i1  ! $i1 = 상위비트끼리 더한 값

        set str3, %o0
        mov %i1, %o1
        mov %i0, %o2
        call printf
        nop

exit:   ret
        restore
```

9[10]. (구현) 한 개의 양의 정수를 십진수 형태로 입력받아 3의 배수일 경우 "mod 3 = 0", 3의 배수 더하기 1일 경우 "mod 3 = 1", 3의 배수 더하기 2일 경우 "mod 3 = 2" 메시지를 출력하는 프로그램을 작성하시오. 또한, 작성한 코드(mid_prob9.s)를 submit하시오. (명령: submit konwoo mid_09)

※ 제한사항: 조건 검사와 내용을 분리하는 switch문 형식으로 구현. bne 사용하지 말 것.

```
bash $ ./mid_prob9
Value?> 360
mod 3 = 0
```

```
bash $ ./mid_prob9
Value?> 211
mod 3 = 1
```

```
bash $ ./mid_prob9
Value?> 3333335
mod 3 = 2
```

!! 답안을 여기에 적으시오. 공간이 부족할 경우, 2열(2-column)로 작성 또는 추가페이지 사용 가능.

```
 1 .section ".text"
 2 str1:   .asciz "Value?> "
 3 str2:   .asciz "%d"
 4 str3:   .asciz "mod 3 = %d"
 5 .align  4
 6 .global main, scanf, printf
 7
 8 main:   save %sp, -96, %sp
 9         mov %g0, %l0          ! int a = 0
10
11 input:  set str1, %o0         ! print "Value?> " for a
12         call printf
13         nop
14
15         set str2, %o0         ! input a
16         add %fp, -4, %o1
17         call scanf
18         nop
19
20         ld [%fp-4], %l0       ! return 0 if a <= 0
21         subcc %l0, 0, %g0
22         ble exit
23         nop
24
25         mov %l0, %o0
26         mov 3, %o1
27         call .urem
28         nop
29         mov %o0, %l1          ! b = a % 3
30
31         subcc %l1, 0, %g0
32         be result
33         nop
34
35         subcc %l1, 1, %g0
36         be result
37         nop
38
39         subcc %l1, 2, %g0
40         be result
41         nop
42
43 result: set str3, %o0
44         mov %l1, %o1
45         call printf
46         nop
47
48 exit:   ret
49         restore
```

10[10]. (구현) 9번과 동일한 문제를 if-else 형식으로 작성하시오. 또한, 작성한 코드(mid_prob10.s)를 submit 하시오. (명령: submit konwoo mid_10)

※ 제한사항: if-else를 통한 구현. be 사용하지 말 것.

```
bash $ ./mid_prob10
Value?> 360
mod 3 = 0
```

```
bash $ ./mid_prob10
Value?> 211
mod 3 = 1
```

```
bash $ ./mid_prob10
Value?> 3333335
mod 3 = 2
```

!! 답안을 여기에 적으시오. 공간이 부족할 경우, 2열(2-column)로 작성 또는 추가페이지 사용 가능.
!! printf가 3개의 인자를 필요로 할 경우, %o0, %o1, %o2를 활용.

```
1  .section ".text"
2  str1:   .asciz "Value?> "
3  str2:   .asciz "%d"
4  str3:   .asciz "mod 3 = %d\n"
5  .align  4
6  .global main, scanf, printf
7
8  main:   save %sp, -96, %sp
9          mov %g0, %l0
10
11 input:  set str1, %o0        ! print "Value?> " for a
12         call printf
13         nop
14
15         set str2, %o0        ! input a
16         add %fp, -4, %o1
17         call scanf
18         nop
19
20         ld [%fp-4], %l0      ! return 0 if a <= 0
21         subcc %l0, 0, %g0
22         ble exit
23         nop
24
25         mov %l0, %o0
26         mov 3, %o1
27         call .urem
28         nop
29         mov %o0, %l1         ! b = a % 3
30
31 mod_0:  subcc %l1, 0, %g0    ! if b != 0
32         bne mod_1            ! jump to "mod_1" label
33         nop
34
35         ba result            ! if b == 0, print result of it
36         nop
37
38 mod_1:  subcc %l1, 1, %g0
39         bne mod_2
40
41         ba result
42         nop
43
44 mod_2:  subcc %l1, 2, %g0
45         bne result
46         nop
47
48 result: set str3, %o0
49         mov %l1, %o1
50         call printf
51         nop
52
53 exit:   ret
54         restore
```