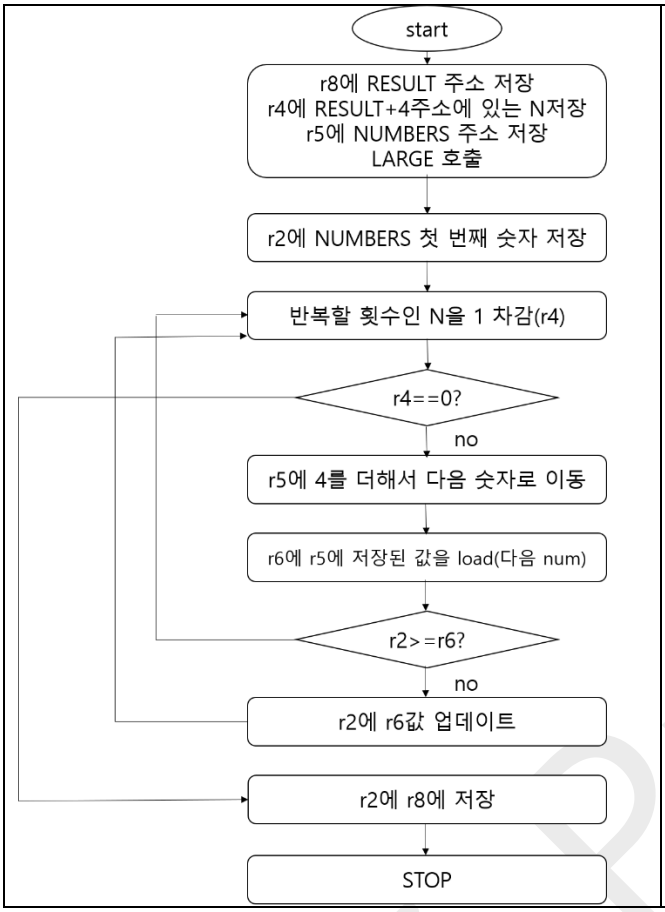
전자 HW 설계 – 실습 보고서

이름: 양해찬 (2016124145)

✓ Part III

동작 원리



NUMBERS 에 저장된 숫자들 중 가장 큰 숫자를 찾는 코드이고,
r2에 초기값으로 NUMBERS의 첫 번째 숫자를 저장 한 후,
다음 숫자와 비교해서 다음 숫자가 크다면 r2에 업데이트 하고 그렇지 않다면 계속 loop를 돈다.
Loop을 돌다가 NUMBERS에 저장된 숫자들의 개수인 N만큼 loop를 돌게 되면 모든 숫자들과 비교를 한 상태이므로 r2값을 리턴한다.

구현 코드 설명

```
            .text
        .global _start

_start:
        movia   r8, RESULT      #r8 -> result 주소
        ldw     r4, 4(r8)       #r4에  r8+4에 있는(N)불러옴
        addi    r5, r8, 8       #r5 -> NUMBERS 주소
        call      LARGE         #LARGE 호출
        stw     r2, (r8)        #r2값을 r8에 저장
        ldw     r7, (r8)        #결과를 확인하기 위해 넣음

STOP:     br      STOP

LARGE:    ldw     r2,     (r5)            #r2에 r5값을 load
LOOP:     subi    r4,     r4,     1       #for와 같은역할
```

```
            beq     r4,     r0,     DONE        #N만큼 반복했으면 return한다.
            addi    r5,     r5,     4           #다음숫자로 이동
            ldw     r6,     (r5)                #r6에 r5의 값을 load
            bge     r2,     r6,     LOOP        #r2값이 r6보다 크면 LOOP로 이동
            mov     r2,     r6                  #r2=r6
            br      LOOP
DONE:       ret                                 #return address로 돌아감

RESULT:     .skip   4                           #저장할 공간
N:          .word   7                           #숫자의 개수
NUMBERS:    .word   4, 5, 3, 6                  #숫자 4개 단위로 저장
            .word   1, 8, 2

            .end
```

Registers

Reg	Value
pc	0x0000001C
zero	0x00000000
r1	0x00000000
r2	0x00000008
r3	0x00000000
r4	0x00000000
r5	0x00000064
r6	0x00000002
r7	0x00000008
r8	0x00000044
r9	0x00000000
r10	0x00000000
r11	0x00000000
r12	0x00000000
r13	0x00000000
r14	0x00000000
r15	0x00000000
r16	0x00000000
r17	0x00000000
r18	0x00000000
r19	0x00000000
r20	0x00000000
r21	0x00000000
r22	0x00000000
r23	0x00000000
et	0x00000000
bt	0xFFFFFFFF
gp	0x00000000
sp	0x00000000
fp	0x00000000
ea	0x00000000
ba	0xFFFFFFFF
ra	0x00000014
status	0x00000000
estatus	0x00000000
bstatus	0xFFFFFFFF
ienable	0x00000000
ipending	0x00000000
cpuid	0x00000000

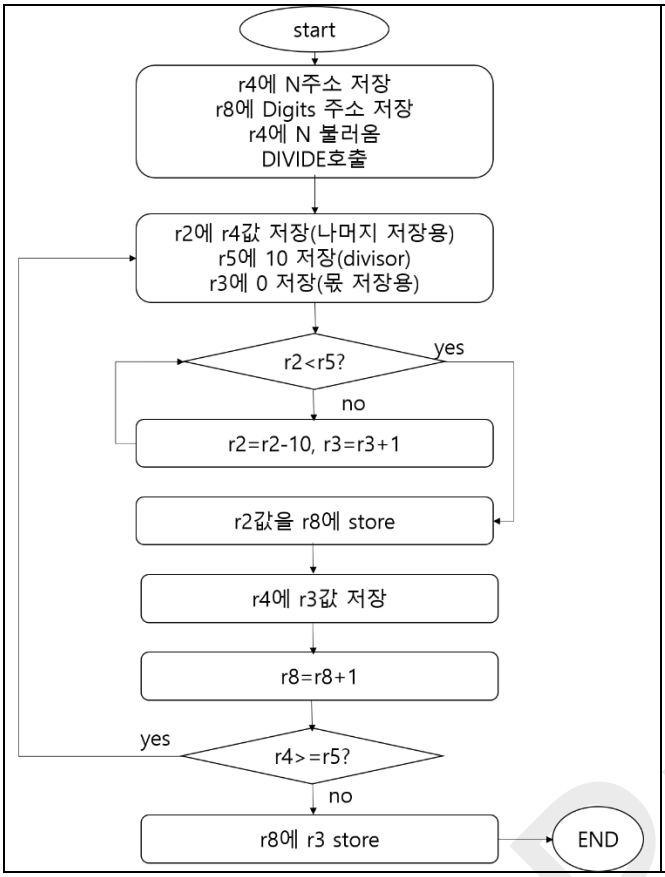
r2에 NUMBERS 중 가장 큰 값인 8이 저장되어 있다.

r6에 NUMBERS 마지막 숫자인 2가 저장되어 있다.
r7에 r8에 저장된 값을 load 했으므로 r2와 같은 값인 8이 저장되어 있다.

전자 HW 설계 – 실습 보고서

✓ Part IV

동작 원리



Binary number를 4decimal digits로 변환하는 코드이다.
즉, 4자리 십진수를 1바이트당 1자리씩 저장하는 코드로,
먼저 r4에 N의 주소 값을 저장하고 r8에 Digits 주소를 저장한다.
r4에 N을 불러오고, divide를 호출해서 10으로 나눠주는데,
10을 한번 뺄 때 마다 몫에 1을 더하는 방식으로 실행한다.
몫은 r3에 저장되고 r2에 나머지가 저장된다. r2가 10미만이면 값을 리턴 한다. N에 1234가 저장 되어 있다고 가정하면 r3에 123, r2에 4가 저장된다. r2값을 r8에 store 하고 그 다음자리를 저장하기 위해 r8의 주소 값에 1을 더해준다. 그리고 r3에 저장되어 있던 123값을 r4에 옮기고, r4가 10보다 크면 계속 반복하게 된다.
10보다 작다면 r3값이 r8에 store된다.

구현 코드 설명

구현 코드와 이에 대한 내용을 설명. 코드를 그대로 가져다 붙이는 것은 지양함. 중요 부분을 자세한 코멘트를 첨가하여 설명.

```
        .text
        .global _start

_start:
        movia   r4,     N               #r4에 N 주소값
        addi    r8,     r4,     4       #r8에 digits 주소값 저장
        ldw     r4,     (r4)            #r4에 N load
        mov     r9,     r8              #r9에 r8의 주소값을 복사해 놓는다
LOOP:
        call        DIVIDE              #DIVIDE호출
        stb     r2,     (r8)            #반환된 r2값 r8에 store
        mov     r4,     r3              #다음 차례에 r2에 넣을 값을 r4에 저장해 놓는다.
        addi    r8,     r8,     1       #다음 자리를 저장하기 위해 주소값 +1
        bge     r4,     r5,     LOOP    #r4가 10보다 크다면 반복
```

```
        stb     r3,     (r8)            #r3를 r8에 store
        ldw     r7,     (r8)            #결과값을 확인하기 위해 넣음

END:    br      END

DIVIDE:
        mov     r2,     r4  #r2=나머지
        movi    r5,     10  #r5=divisor
        movi    r3,     0   #r3=몫
CONT:
        blt     r2,     r5,     DIV_END #r2<r5면 종료(나눌 숫자가 10미만이면 종료)
        sub     r2,     r2,     r5      #r2=r2-10
        addi    r3,     r3,     1       #몫 계산
        br      CONT
DIV_END:    ret

N:          .word   1234
Digits:     .space  4

        .end
```

Registers

Reg	Value
pc	0x00000030
zero	0x00000000
r1	0x00000000
r2	0x00000002
r3	0x00000001
r4	0x00000001
r5	0x0000000A
r6	0x00000000
r7	0x01020304
r8	0x0000005B
r9	0x00000058
r10	0x00000000
r11	0x00000000
r12	0x00000000
r13	0x00000000
r14	0x00000000
r15	0x00000000
r16	0x00000000
r17	0x00000000
r18	0x00000000
r19	0x00000000
r20	0x00000000
r21	0x00000000
r22	0x00000000
r23	0x00000000
et	0x00000000
bt	0xFFFFFFFF
gp	0x00000000
sp	0x00000000
fp	0x00000000
ea	0x00000000
ba	0xFFFFFFFF
ra	0x00000018
status	0x00000000
estatus	0x00000000
bstatus	0xFFFFFFFF
ienable	0x00000000
ipending	0x00000000
cpuid	0x00000000

r2는 마지막으로 DIVIDE가 실행되고 난 후 나머지 이므로
1234입력->몫 123, 나머지 4->몫 12, 나머지 3->몫 1, 나머지 2 따라서 2가 저장된다.
r3는 마지막 DIVIDE가 실행되고 난 후 몫이 저장되므로 1이 저장된다.
r4는 r3과 같으므로 1이 저장된다.
r5는 divisor으로, 10이 저장되어 있다.
r7에 원했던 결과인 0x01020304가 저장되어 있다.
r8은 r8의 초기 주소 값인 r9보다 3이 커야 한다. (loop 3번 실행)
r9 = 0x01011000 r8 = 0x01011011