어셈블리프로그래밍설계 및 실습 보고서

실험제목: Pseudo Instructions

실험일자: 2023년 10월 31일 (화)

제출일자: 2023년 11월 8일 (수)

학 과: 컴퓨터공학과

담당교수: 이준환 교수님

실습분반: 화 6,7 목 5

학 번: 2022202065

성 명: 박나림

1. 제목 및 목적 (3%)

A. 제목

Pseudo Instructions

B. 목적

Pseudo instructions들을 공부하고 이들이 실제 instruction으로 어떻게 변환되는지 확인하도록 한다. 코드를 통해 Disassembly를 해석하는 방법을 이해할 수 있도록 한다. 이러한 과정에서 사용되는 Label의 이름들이 실제 메모리 주소로 사용되는 것을 이해하고 코드를 설계한다.

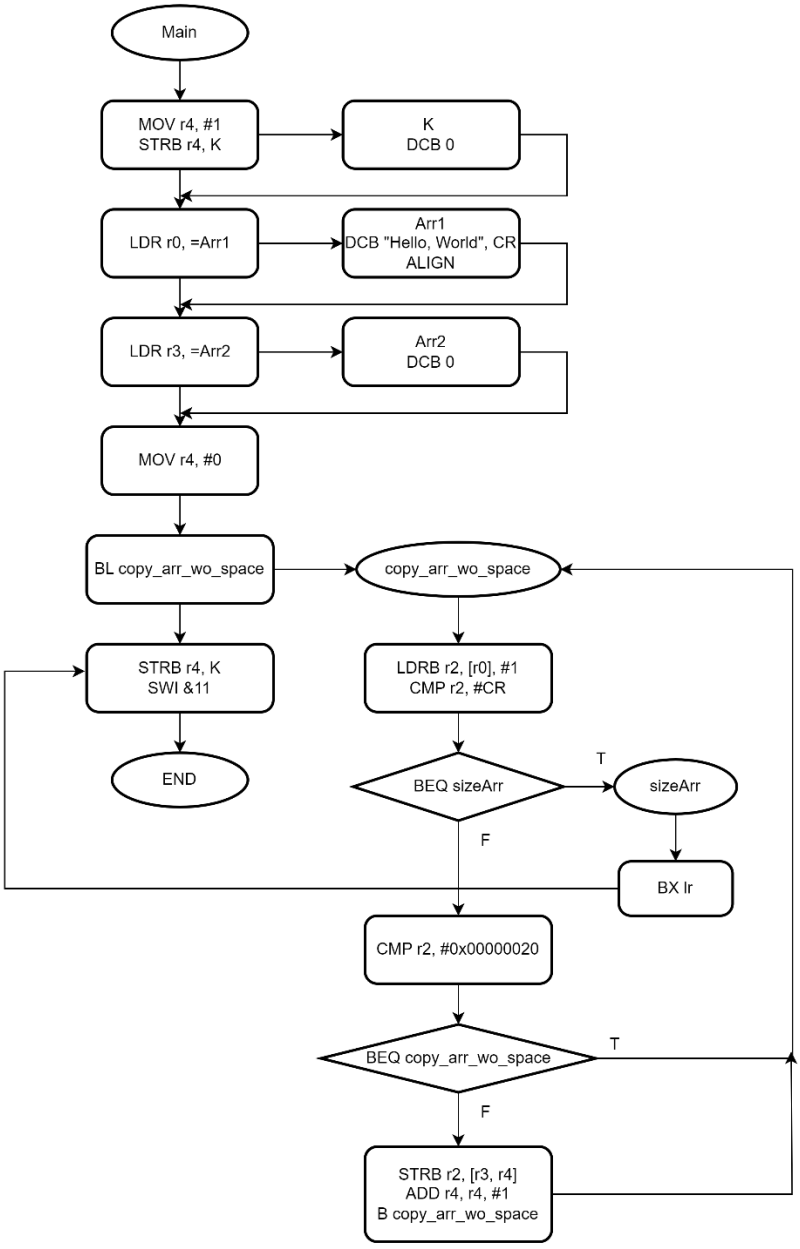
2. 설계 (Design) (50%)

A. Pseudo code

```
CR EQU 0x0D
 AREA ARMex, CODE, READONLY
            ENTRY
Main
        MOV r4, #1 ;K값
        STRB r4, K ;K값 초기화
        LDR r0, =Arr1 ;복사할 문자열 저장된 주소 불러옴
        LDR r3, =Arr2 ;문자열 저장될 주소 불러옴
        MOV r4, #0 ;
        BL copy_arr_wo_space

        STRB r4, K ;size of Arr2
        SWI &11 ;finish

copy_arr_wo_space
        LDRB r2,[r0],#1 ;byte단위로 문자 하나씩 불러오면서 업데이트 시킴
        CMP r2,#CR ;문자열 끝이면
        BEQ sizeArr ;반환
        CMP r2, #0x00000020 ;space면
        BEQ copy_arr_wo_space ;continue

        STRB r2,[r3,r4] ;byte단위로 index값에 따라 문자 하나씩 저장함
        ADD r4, r4, #1 ;index 업데이트
        B copy_arr_wo_space ;문자열 끝까지 반복

sizeArr
        BX lr ;함수 반환
```

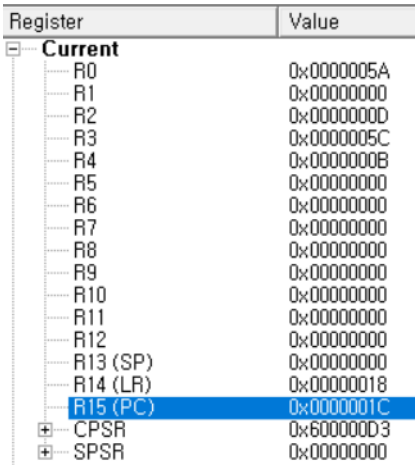
```
AREA dataArray, DATA ;Data area
K
        DCB 0
Arr1
        DCB "Hello, World",CR
        ALIGN
Arr2
        DCB 0

        END
```

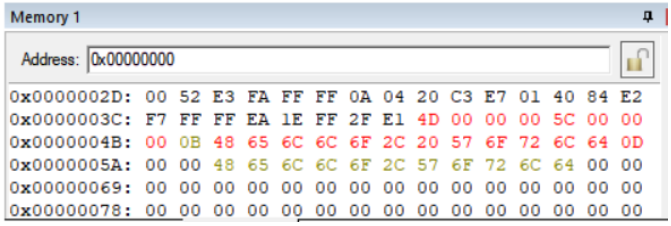
B. Flow chart 작성



C. Result

Arr1의 주소가 4D로 할당되었고, Arr2의 주소는 5C, K의 주소는 4C로 할당된 것을 볼 수 있다. 이후 4D부터 저장된 문자를 하나씩 불러와서 5C부터 차례대로 공백과 문자열 끝(CR) 제외 저장하며 마지막에 R4로 Arr2의 size값인 B를 4C에 저장한 모습이다.





D. Performance

-code size: 76

```
Program Size: Code=76
".\Objects\test3.axf"
Build Time Elapsed:  0
```

-state: 177

3. 고찰 및 결론

A. 고찰 (35%)

DCB로 문자열들의 메모리 주소를 할당해준 뒤 STRB로 메모리에 저장하려고 하니 에러가 발생하였다. 확인해보니 메모리 영역이 서로 겹쳐서 발생한 문제여서, Linker설정에서 R/W base값을 0x0에서 에러가 난 시점부터로 바꾸니 정상적으로 실행될 수 있었다.

또한 라인 별로 코드를 실행할 때 STRB부분에서 4byte까지만 저장되고 그 다음 문자부터는 메모리에 쓸 수가 없다는 오류가 발생하였다. 이는 memory.ini 파일에서 MAP의 시작 주소를 문자열이 저장되는 메모리의 주소로 바꾸니 문자열이 전부 끝까지 저장될 수 있었다.

기존에는 문자열을 선언하고 그 주소까지 저장하여 사용했었는데 pseudo instruction을 사용하니 바로 처리될 수 있어서 register 사용량은 더 줄은 것 같다. 하지만 이 역시 컴파일러가 직접 다른 명령어들로 처리하는 것이기 때문에 그 점을 고려하여 잘 사용해야겠다는 생각이 들었다.

B. 결론 (10%)

LDR을 사용할 때 선언된 label 이름을 바로 주소로 사용하니 더 알아보기 편하다고 느꼈다. 또한 이렇게 2개 이상의 DCB를 선언할 때 메모리 영역이 겹치면 R/W 주소 값을 바꿈으로써 해결할 수 있다는 점을 깨달았다. 이와 관련하여, code size를 확인하려고 할 때 값이 나오지 않는다면 art+f7로 확인할 수 있다는 점도 알게 되었다.

이번에 직접 메모리 주소에 복사한 문자열을 차례대로 저장하면서 메모리를 최대한 낭비하지 않고 사용하는 방법에 대해 생각해보게 되었다. 다른 프로그램을 구현할 때도 직접 메모리 주소를 설정해주어서 메모리를 효율적으로 사용하는 식으로 응용해볼 수 있을 것 같다.

4. 참고문헌 (2%)

이준환 / 어셈블리프로그램설계및실습 강의자료 / 광운대학교(컴퓨터공학과) / 2023