Assignment #4

RISC-V Assembly Programming

1. 개요

본 과제에서는 RISC-V ISA 를 사용하여 정렬 알고리즘 중 하나인 Bubble Sort 를 구현한다. 구현한 프로그램은 앞으로 구현할 RISC-V 프로세서의 테스트 프로그램으로 활용한다.

2. RISC-V ASSEMBLY PROGRAMMING

RISC-V ISA 를 활용한 프로그래밍 과정을 소개한다. 본 과제에서는 구현의 편의성을 위해 온라인으로 제공되는 시뮬레이션 환경을 사용한다. 시뮬레이터는 다음 링크를 통해 접속한다.

시뮬레이터 링크: https://ascslab.org/research/briscv/simulator/simulator.html

주어진 시뮬레이터는 32-bit 크기의 레지스터를 사용하는 RV32 시스템을 지원한다. Assembly Program 의 구조는 코드 정의하기 위한 .text 영역과 데이터를 정의하기 위한 .data 영역, 그리고 읽기 전용 데이터를 정의하기 위한 .rodata 영역으로 나뉜다.

.text 영역

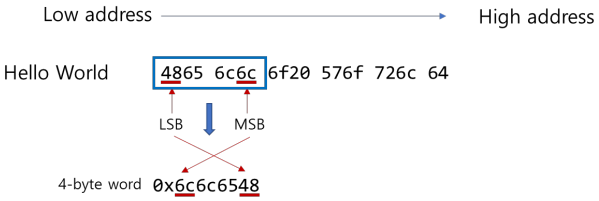
.text 영역은 프로그램의 코드가 정의되는 영역으로 .text 영역 아래 어셈블리 코드를 작성하게 된다. 다음은 .text 영역을 정의하고 함수를 구현하는 예시이다. C 코드에서 정의하는 함수의 이름은 함수의 첫 명령어의 위치를 가리키는 위치에 불과하며, 어셈블리 명령에서 첫 명령어의 레이블로 표현한다.

```
.text                    # #으로 시작되는 문장은 주석입니다.
                         # .text directive 아래는 .text 영역으로 정의됩니다.
.align 2                 # 아래 명령어를 메모리에 배치 시 4 byte 단위로 배치합니다.
.globl main              # main 함수를 외부의 파일이 볼 수 있도록 알려주는 역할입니다.
                         # 함수가 해당 파일에서만 사용되는 local 함수일 경우에는 불필요합니다.
main:                    # main 함수의 위치를 나타냅니다. 첫 명령어의 주소를 가리킵니다.
        addi t0, zero, 1
         …
        jarl zero, ra
```

.data/.rodata 영역

.data 와 .rodata 영역은 각각 읽고 쓰기가 가능한 데이터 영역과 읽기만 가능한 데이터 영역을 나타낸다. .text 영역과 동일하게 각 영역을 나타내는 directive 를 사용하여 정의된다. 다음의 .data 와 .rodata 영역에 int array[5] = {1, 2, 3, 4, 5}와 const char msg[] = "Hello World"를 정의하는 예시이다.

```
.data
.align 2
array:
	.word 1 # 각 데이터는 .word directive 를 사용하여 정의한다.
	.word 2
	.word 3
	.word 0x4 # 16 진수 표현도 지원함.
	.word 0x5
.rodata
msg:
	# Hello World 의 ASCII 코드 값(16 진수, HEX)은 다음과 같다.
	# 4865 6c6c 6f20 576f 726c 64
	# .word 로 데이터를 표현하기 위해서는 4 byte 단위로 묶어서 표시하여야 한다.
	# RISC-V 는 little-endian 시스템이기 때문에 4 byte 의 LSB 가 먼저 저장되므로,
	# 문자를 순서대로 저장하기 위해서는 4 byte 로 묶을 때 순서를 뒤집어서 표시해야 한다.
```




```
	# 0x48656c6c 가 아니라 0x6c6c6548 로 표현해야 한다.
	.word  0x6c6c6548      # 0x48656c6c
	.word  0x6f57206f      # 0x6f20576f
	.word  0x00646c72      # 0x726c6400
```

RISC-V 레지스터

시뮬레이터의 어셈블러는 RISC-V 레지스터 사용 시, 레지스터명 x[num]을 지원하지 않고, Abstract Binary Interface (ABI)에서 정의한 이름만 지원한다. (ABI 는 컴파일된 바이너리 간에 함수 등을 공유하기 위한 표준 규약). 각 레지스터의 ABI 이름은 다음과 같이 정의되어 있다

RISC-V Calling Convention			
Register	ABI Name	Saver	Description
x0	zero	---	Hard-wired zero
x1	ra	Caller	Return address
x2	sp	Callee	Stack pointer
x3	gp	---	Global pointer
x4	tp	---	Thread pointer
x5-7	t0-2	Caller	Temporaries
x8	s0/fp	Callee	Saved register/frame pointer
x9	s1	Callee	Saved register
x10-11	a0-1	Caller	Function arguments/return values
x12-17	a2-7	Caller	Function arguments
x18-27	s2-11	Callee	Saved registers
x28-31	t3-t6	Caller	Temporaries
f0-7	ft0-7	Caller	FP temporaries
f8-9	fs0-1	Callee	FP saved registers
f10-11	fa0-1	Caller	FP arguments/return values
f12-17	fa2-7	Caller	FP arguments
f18-27	fs2-11	Callee	FP saved registers
f28-31	ft8-11	Caller	FP temporaries

RISC-V 명령어

강의자료 및 첨부된 riscv-card.pdf 참조

3. BUBBLE SORT

Bubble Sort 알고리즘은 간단한 정렬 알고리즘 중의 하나로, 주어진 배열의 인접한 원소를 비교 정렬하는 알고리즘이다. Bubble Sort 는 연산량은 $O(n^2)$으로 비효율적이기 때문에 실제 응용프로그램에서 사용되지는 않는다. 본 과제에서는 RISC-V ISA 를 활용한 프로그래밍 연습 목적으로 사용한다.

다음은 Bubble Sort 알고리즘의 동작 예시이다. 순서대로 5, 1, 4, 2, 8 숫자를 가진 배열이 주어졌다고 가정하자. [출처: Wikipedia]

First Pass

(**5 1** 4 2 8) → (**1 5** 4 2 8) 첫째, 둘째 원소 5, 1 의 대소를 비교한다. 첫째 원소 5 가 더 크기 때문에 두 원소의 위치를 변경한다.

(1 **5 4** 2 8) → (1 **4 5** 2 8) 선택된 5, 4 를 정렬한다.

(1 4 **5 2** 8) → (1 4 **2 5** 8) 선택된 5, 2 를 정렬한다.

(1 4 2 **5 8**) → (1 4 2 **5 8**) 이미 정렬되어 있기에 다음 단계로 넘어간다.

Second Pass

(**1 4** 2 5 8) → (**1 4** 2 5 8) 이미 정렬되어 있다.

(1 **4 2** 5 8) → (1 **2 4** 5 8) 선택된 4, 2 를 정렬한다.

(1 2 **4 5** 8) → (1 2 **4 5** 8) 이미 정렬되어 있다.

(1 2 4 **5 8**) → (1 2 4 **5 8**) 이미 정렬되어 있다.

아직 정렬이 완료되었는지 알 수 없기 때문에 한 번 더 실행한다.

Third Pass

(**1 2** 4 5 8) → (**1 2** 4 5 8)

(1 **2 4** 5 8) → (1 **2 4** 5 8)

(1 2 **4 5** 8) → (1 2 **4 5** 8)

(1 2 4 **5 8**) → (1 2 4 **5 8**)

인접한 원소 간에 정렬이 발생하지 않았기 때문에 배열 정렬이 완료되었다.

구현에 도움을 주기 위해 bubblesort.s 템플릿을 제공한다. 템플릿을 실행하면 시뮬레이터의 콘솔에 정렬 전 배열의 값과 정렬 후 배열의 값이 출력된다. 템플릿의 비어있는 bubblesort 함수를 다음 Pseudocode 를 참고하여 구현하여, 정렬 후 배열 값이 정상 출력되도록 수정하여라.

```
procedure bubbleSort(A : list of sortable items) [출처: Wikipedia]
    n := length(A)
    repeat
        swapped := false
        for i := 1 to n-1 inclusive do
            /* if this pair is out of order */
            if A[i-1] > A[i] then
                /* swap them and remember something changed */
                swap(A[i-1], A[i])
                swapped := true
            end if
        end for
    until not swapped
end procedure
```

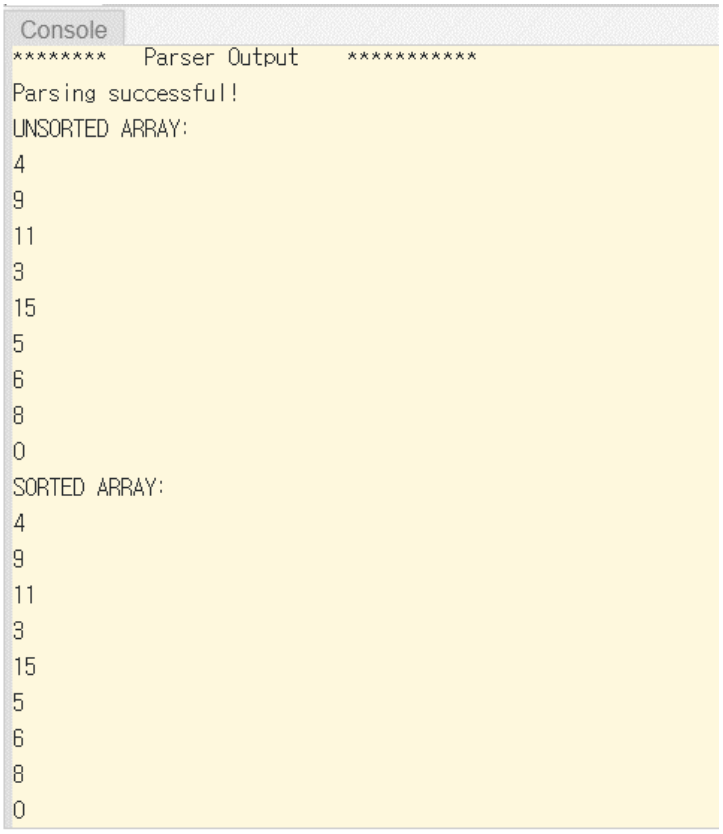


그림 1. 템플릿 실행 결과

Bubblesort 함수가 구현되지 않았기에 정렬 후 배열의 출력 결과가 정렬되어 있지 않다.