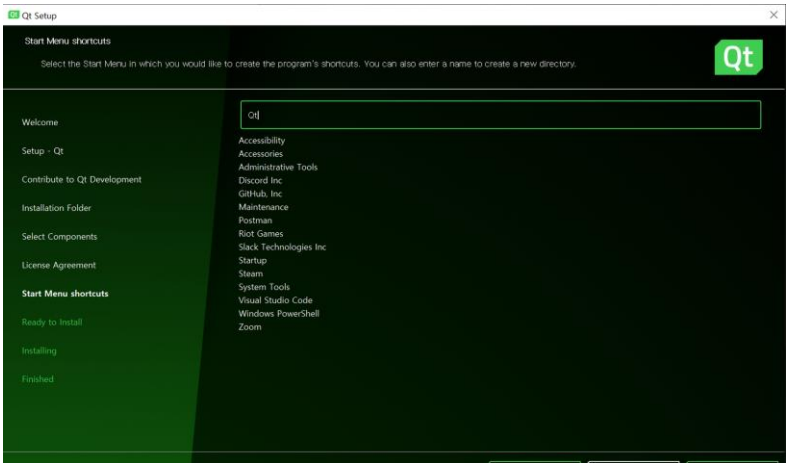
과목명	컴퓨터 구조
담당교수	송인식 교수님
학과	소프트웨어학과
학번	32200185
이름	곽다은
제출일자	2021.10.24

Selection sorting with RISC-V

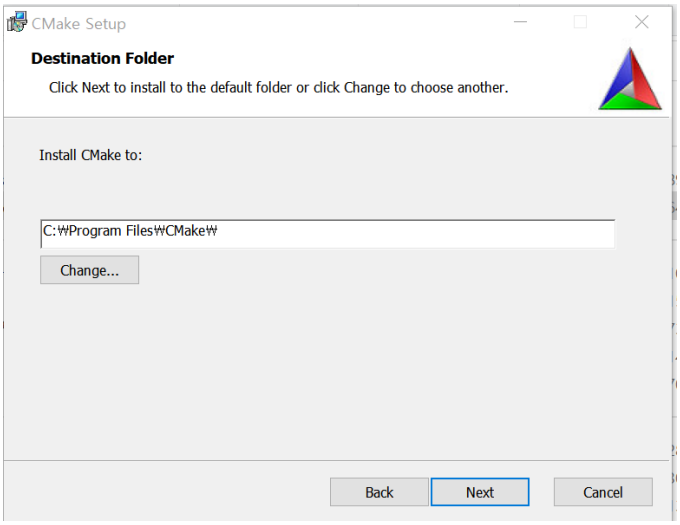
1. RISC-V Simulator Configuration

많은 종류의 RISC-V 시뮬레이터 중에서 Ripes를 선택하였다. 개발자의 github에서 다운받을 수 있었고, Ripes를 사용하기 전 원활한 컴파일을 위해 프로그램 개발에 널리 쓰이는 크로스 플랫폼 프레임워크 Qt와, 운영체제에 맞게 make파일을 생성 및 관리해주는 Cmake까지 설치해주었다.

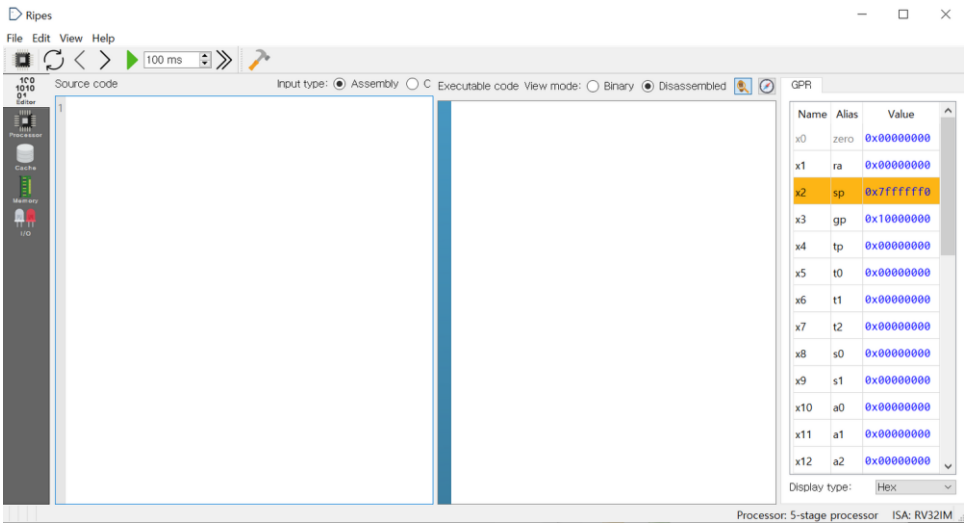
- Qt 설치 후 실행화면



- Cmake 설치화면



- Ripes 설치 후 실행화면



2. Selection sorting

선택 정렬 알고리즘이란, 제자리 정렬 알고리즘에 속하는 알고리즘으로, 정렬되지 않은 입력 배열 이외에 다른 추가 메모리를 요구하지 않는 정렬 방법이다.

1. 주어진 배열 중 최솟값을 찾는다.
2. 찾은 최솟값을 맨 앞에 위치한 값과 교체한다.
3. 맨 처음 위치를 제외한 나머지 배열을 같은 방법으로 교체한다.
4. 하나의 원소만 남을 때까지 위의 과정을 반복한다.

3. RISC-V Assembly Code

```
.data
arr: .word 10, 100, 20, 2000, 300       # 배열을 선언해서 원소들을 담아줌

.text
main:
        la x3, arr                        # gp가 선언한 arr를 가리키도록 함
        addi x5, x0, 0                    # i를 초기화
L1:
        addi x7, x5, 0                    # min = i
        addi x6, x5, 1                    # j = i + 1

L2:
        slli x29, x6, 2
        add x13, x3, x29                  # arr[ i ] address를 shift연산된 값으로 교체
        lw x10, 0(x13)                    # 저장

        slli x30, x7, 2
        add x14, x3, x30                  # arr[ j ] address를 shift연산된 값으로 교체
        lw x11, 0(x14)                    # 저장

        bge x10, x11, nochange            # 크기비교 -> nochange
        addi x7, x6, 0                    # min값 업데이트

nochange:
        addi x6, x6, 1                    # j = j + 1
        addi x19, x0, 5
        blt x6, x19, L2                   # j, j condition을 비교 -> 루프 종료 or 다시 루프로 돌아감

        slli x28, x5, 2
        add x12, x3, x28                  # shift연산된 결과를 address에 집어넣어 원소에 접근 가능하
도록 함

        slli x30, x7, 2
        add x14, x3, x30                  # shift연산된 결과를 address에 집어넣어 최솟값에 접근 가능하도록
함
```

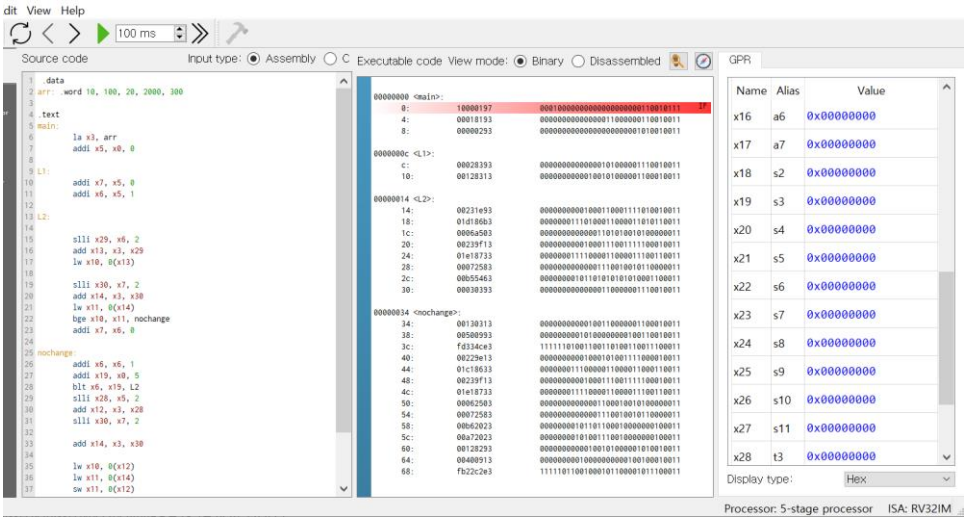
```
lw x10, 0(x12)              # temp = arr[ i ]
lw x11, 0(x14)              # arr[ i ] = arr[ min ]
sw x11, 0(x12)
sw x10, 0(x14)              # arr[ min ] = temp

addi x5, x5, 1
addi x18, x0, 4
blt x5, x18, L1
```

main : 선언한 배열의 base address를 가져온다.

L1 : 정렬을 위한 큰 루프로, 최솟값을 저장하고 L2를 관리한다.

L2 : L1 내부의 루프로, 정렬되지 않은 부분에서 최솟값을 찾아 맨 앞의 원소와 바꿔준다.

nochange : 최솟값을 업데이트할 필요가 없는 경우, 이 과정을 건너뛴다.



Ripes에 입력한 결과, 실행가능한 executable Binary code가 만들어졌다.