

Files

main

- 0911Assignment.md
- 0918Assignment.md
- 0925Assignment.md
- 1016Assignment.md
- 1030Assignment.md
- 1106Assignment.md
- 1113Assignment.md
- 1120Assignment.md
- 1127Assignment.md
- 1211FinalAssignment.md
- 1211Presentation
- Readme.md
- class0911.md
- class0925.md
- class1016.md
- class1103.md
- class1106.md
- class1120.md
- class1127.md
- 그레이프카드 발표.pdf
- README.md

READMD.md

Assembly2025 / Report / 1211FinalAssignment.md

Lwy1214 Update 1211FinalAssignment.md

11ac02f · last week · History

Preview Code Blame Raw

Copilot gpt5 사용

1) 어셈블리 + C 코드를 ArraySum으로 만들어보기

Ubuntu WSL 환경 사용

실행 환경 준비 (Ubuntu/WSL)

1. 필요한 패키지 설치

```
sudo apt update
sudo apt install build-essential gcc-multilib
```

2. 컴파일 방법

- o 32비트 모드로 빌드:

```
gcc -m32 test.c -o test
```

- o 64비트 모드로 빌드:

```
gcc test.c -o test
```

→ 어떤 모드를 선택할지는 코드에서 사용하는 레지스터 이름에 따라 달라집니다.

지금은 64비트 환경이 기본이므로 rax, rsi, rcx를 쓰는 버전을 보여드릴게요.

★ 수정된 64비트 버전 (AT&T 문법)

```
#include <stdio.h>

int ArraySum_GCC(int array[], int count) {
    int sum = 0;

    __asm__ volatile (
        "movl $0, %%eax\n\t"           // sum = 0
        "test %%rcx, %%rcx\n\t"        // count == 0 ?
        "jne end_loop\n\t"
        "loop_start:\n\t"
        "addl (%rsi), %%eax\n\t"       // eax += *rsi
        "addq $4, %%rsi\n\t"           // rs1 += 4 (int 크기)
        "loop loop_start\n\t"
        "end_loop:\n\t"
        : "+a" (sum)                // 출력: eax += sum
        : "S" (array), "c" (count)   // 입력: rs1=array, rcx=count
        : "memory"
    );

    return sum;
}

int main() {
    int test_array[] = {10, 20, 30, 40, 50};
    int array_count = 5;

    int result_sum = ArraySum_GCC(test_array, array_count);
    printf("Array Sum (GCC style): %d\n", result_sum);

    return 0;
}
```

★ 실행 예시

gcc test.c -o test

출작:

Array Sum (GCC style): 150

이렇게 하면 WSL/Ubuntu 64비트 환경에서 정상적으로 실행됩니다.

혹시 원하시면 제거 32비트 버전도 같이 만들어 드릴 수 있어요.

```
lwy1214@LEGION540-Assembl~:~+~+
[GNOME-nano 6.2] Arraysum.c ~
#include <stdio.h>

int ArraySum_GCC(int array[], int count) {
    int sum = 0;

    __asm__ volatile (
        "movl $0, %%eax\n\t"           // sum = 0
        "test %%rcx, %%rcx\n\t"        // count == 0 ?
        "jne end_loop\n\t"
        "loop_start:\n\t"
        "addl (%rsi), %%eax\n\t"       // eax += *rsi
        "addq $4, %%rsi\n\t"           // rs1 += 4 (int 크기)
        "loop loop_start\n\t"
        "end_loop:\n\t"
        : "+a" (sum)                // 출력: eax += sum
        : "S" (array), "c" (count)   // 입력: rs1=array, rcx=count
        : "memory"
    );

    return sum;
}

int main() {
    int test_array[] = {10, 20, 30, 40, 50};
    int array_count = 5;

    int result_sum = ArraySum_GCC(test_array, array_count);
    printf("Array Sum (GCC style): %d\n", result_sum);

    return 0;
}
```

★ 실행 흐름 예시

배열: {10, 20, 30, 40, 50}

- 루프 1: rax = 0 + 10
- 루프 2: rax = 10 + 20 = 30
- 루프 3: rax = 30 + 30 = 60
- 루프 4: rax = 60 + 40 = 100
- 루프 5: rax = 100 + 50 = 150 → 최종 결과: 150

○ 오류

- ArraySum_GCC는 배열 주소(rs1)와 길이(rcx)를 데지스터에 넣고,
- rax에 합계를 누적하면서 루프를 끝냅니다.
- 루프는 loop 명령으로 rcx를 자동 감소시키며 반복합니다.
- 결과는 rax = sum으로 반환됩니다.

2) 라이브러리

1. 정적 라이브러리 (Static Library, .a)

정적 라이브러리는 컴파일 시점에 프로그램에 포함됩니다.

단계

1. 소스 파일 분리

o arraysum.c :

```
int ArraySum_GCC(int array[], int count) {
    int sum = 0;
```

__asm__ volatile :

```
"movl $0, %%eax\n\t"           // sum = 0
"test %%rcx, %%rcx\n\t"        // count == 0 ?
"jne end_loop\n\t"
"loop_start:\n\t"
"addl (%rsi), %%eax\n\t"       // eax += *rsi
"addq $4, %%rsi\n\t"           // rs1 += 4 (int 크기)
"loop loop_start\n\t"
"end_loop:\n\t"
: "+a" (sum)                // 출력: eax += sum
: "S" (array), "c" (count)   // 입력: rs1=array, rcx=count
: "memory"
```

);

return sum;

2. 오브젝트 파일 생성

gcc -c arraysum.c -o arraysum.o

3. 라이브러리 묶기

ar rcs libarraysum.a arraysum.o

→ libarraysum.a라는 정적 라이브러리 파일이 생깁니다.

4. 사용하기

o main.c :

```
#include <stdio.h>
#include "arraysum.h"
```

```
int main() {
    int arr[] = {10, 20, 30, 40, 50};
    printf("Sum: %d\n", ArraySum_GCC(arr, 5));
    return 0;
}
```

○ 컴파일:

gcc main.c -L. -larraysum -o main

(. . .은 현재 디렉토리에서 라이브러리를 찾으라는 의미, -larraysum은 libarraysum.a를 링크하라는 의미)

```
lwy1214@LEGION540-Assembl~:~+~+
[GNOME-nano 6.2] Arraysum.c ~
#include <stdio.h>

int ArraySum_GCC(int array[], int count) {
    int sum = 0;

    __asm__ volatile (
        "movl $0, %%eax\n\t"           // sum = 0
        "test %%rcx, %%rcx\n\t"        // count == 0 ?
        "jne end_loop\n\t"
        "loop_start:\n\t"
        "addl (%rsi), %%eax\n\t"       // eax += *rsi
        "addq $4, %%rsi\n\t"           // rs1 += 4 (int 크기)
        "loop loop_start\n\t"
        "end_loop:\n\t"
        : "+a" (sum)                // 출력: eax += sum
        : "S" (array), "c" (count)   // 입력: rs1=array, rcx=count
        : "memory"
    );

    return sum;
}

int main() {
    int test_array[] = {10, 20, 30, 40, 50};
    int array_count = 5;

    int result_sum = ArraySum_GCC(test_array, array_count);
    printf("Array Sum (GCC style): %d\n", result_sum);

    return 0;
}
```

```
lwy1214@LEGION540-Assembl~:~+~+
[GNOME-nano 6.2] Arraysum.c ~
#include <stdio.h>

int ArraySum_GCC(int array[], int count) {
    int sum = 0;

    __asm__ volatile (
        "movl $0, %%eax\n\t"           // sum = 0
        "test %%rcx, %%rcx\n\t"        // count == 0 ?
        "jne end_loop\n\t"
        "loop_start:\n\t"
        "addl (%rsi), %%eax\n\t"       // eax += *rsi
        "addq $4, %%rsi\n\t"           // rs1 += 4 (int 크기)
        "loop loop_start\n\t"
        "end_loop:\n\t"
        : "+a" (sum)                // 출력: eax += sum
        : "S" (array), "c" (count)   // 입력: rs1=array, rcx=count
        : "memory"
    );

    return sum;
}

int main() {
    int test_array[] = {10, 20, 30, 40, 50};
    int array_count = 5;

    int result_sum = ArraySum_GCC(test_array, array_count);
    printf("Array Sum (GCC style): %d\n", result_sum);

    return 0;
}
```

```
lwy1214@LEGION540-Assembl~:~+~+
[GNOME-nano 6.2] Arraysum.c ~
#include <stdio.h>

int ArraySum_GCC(int array[], int count) {
    int sum = 0;

    __asm__ volatile (
        "movl $0, %%eax\n\t"           // sum = 0
        "test %%rcx, %%rcx\n\t"        // count == 0 ?
        "jne end_loop\n\t"
        "loop_start:\n\t"
        "addl (%rsi), %%eax\n\t"       // eax += *rsi
        "addq $4, %%rsi\n\t"           // rs1 += 4 (int 크기)
        "loop loop_start\n\t"
        "end_loop:\n\t"
        : "+a" (sum)                // 출력: eax += sum
        : "S" (array), "c" (count)   // 입력: rs1=array, rcx=count
        : "memory"
    );

    return sum;
}

int main() {
    int test_array[] = {10, 20, 30, 40, 50};
    int array_count = 5;

    int result_sum = ArraySum_GCC(test_array, array_count);
    printf("Array Sum (GCC style): %d\n", result_sum);

    return 0;
}
```

```
lwy1214@LEGION540-Assembl~:~+~+
[GNOME-nano 6.2] Arraysum.c ~
#include <stdio.h>

int ArraySum_GCC(int array[], int count) {
    int sum = 0;

    __asm__ volatile (
        "movl $0, %%eax\n\t"           // sum = 0
        "test %%rcx, %%rcx\n\t"        // count == 0 ?
        "jne end_loop\n\t"
        "loop_start:\n\t"
        "addl (%rsi), %%eax\n\t"       // eax += *rsi
        "addq $4, %%rsi\n\t"           // rs1 += 4 (int 크기)
        "loop loop_start\n\t"
        "end_loop:\n\t"
        : "+a" (sum)                // 출력: eax += sum
        : "S" (array), "c" (count)   // 입력: rs1=array, rcx=count
        : "memory"
    );

    return sum;
}

int main() {
    int test_array[] = {10, 20, 30, 40, 50};
    int array_count = 5;

    int result_sum = ArraySum_GCC(test_array, array_count);
    printf("Array Sum (GCC style): %d\n", result_sum);

    return 0;
}
```

```
lwy1214@LEGION540-Assembl~:~+~+
[GNOME-nano 6.2] Arraysum.c ~
#include <stdio.h>

int ArraySum_GCC(int array[], int count) {
    int sum = 0;

    __asm__ volatile (
        "movl $0, %%eax\n\t"           // sum = 0
        "test %%rcx, %%rcx\n\t"        // count == 0 ?
        "jne end_loop\n\t"
        "loop_start:\n\t"
        "addl (%rsi), %%eax\n\t"       // eax += *rsi
        "addq $4, %%rsi\n\t"           // rs1 += 4 (int 크기)
        "loop loop_start\n\t"
        "end_loop:\n\t"
        : "+a" (sum)                // 출력: eax += sum
        : "S" (array), "c" (count)   // 입력: rs1=array, rcx=count
        : "memory"
    );

    return sum;
}

int main() {
    int test_array[] = {10, 20, 30, 40, 50};
    int array_count = 5;

    int result_sum = ArraySum_GCC(test_array, array_count);
    printf("Array Sum (GCC style): %d\n", result_sum);

    return 0;
}
```

```
lwy1214@LEGION540-Assembl~:~+~+
[GNOME-nano 6.2] Arraysum.c ~
#include <stdio.h>

int ArraySum_GCC(int array[], int count) {
    int sum = 0;

    __asm__ volatile (
        "movl $0, %%eax\n\t"           // sum = 0
        "test %%rcx, %%rcx\n\t"        // count == 0 ?
        "jne end_loop\n\t"
        "loop_start:\n\t"
        "addl (%rsi), %%eax\n\t"       // eax += *rsi
        "addq $4, %%rsi\n\t"           // rs1 += 4 (int 크기)
        "loop loop_start\n\t"
        "end_loop:\n\t"
        : "+a" (sum)                // 출력: eax += sum
        : "S" (array), "c" (count)   // 입력: rs1=array, rcx=count
        : "memory"
    );

    return sum;
}

int main() {
    int test_array[] = {10, 20, 30, 40, 50};
    int array_count = 5;

    int result_sum = ArraySum_GCC(test_array, array_count);
    printf("Array Sum (GCC style): %d\n", result_sum);

    return 0;
}
```

```
lwy1214@LEGION540-Assembl~:~+~+
[GNOME-nano 6.2] Arraysum.c ~
#include <stdio.h>

int ArraySum_GCC(int array[], int count) {
    int sum = 0;

    __asm__ volatile (
        "movl $0, %%eax\n\t"           // sum = 0
        "test %%rcx, %%rcx\n\t"        // count == 0 ?
        "jne end_loop\n\t"
        "loop_start:\n\t"
        "addl (%rsi), %%eax\n\t"       // eax += *rsi
        "addq $4, %%rsi\n\t"           // rs1 += 4 (int 크기)
        "loop loop_start\n\t"
        "end_loop:\n\t"
        : "+a" (sum)                // 출력: eax += sum
        : "S" (array), "c" (count)   // 입력: rs1=array, rcx=count
        : "memory"
    );

    return sum;
}

int main() {
    int test_array[] = {10, 20, 30, 40, 50};
    int array_count = 5;

    int result_sum = ArraySum_GCC(test_array, array_count);
    printf("Array Sum (GCC style): %d\n", result_sum);

    return 0;
}
```

```
lwy1214@LEGION540-Assembl~:~+~+
[GNOME-nano 6.2] Arraysum.c ~
#include <stdio.h>

int ArraySum_GCC(int array[], int count) {
    int sum = 0;

    __asm__ volatile (
        "movl $0, %%eax\n\t"           // sum = 0
        "test %%rcx, %%rcx\n\t"        // count == 0 ?
        "jne end_loop\n\t"
        "loop_start:\n\t"
        "addl (%rsi), %%eax\n\t"       // eax += *rsi
        "addq $4, %%rsi\n\t"           // rs1 += 4 (int 크기)
        "loop loop_start\n\t"
        "end_loop:\n\t"
        : "+a" (sum)                // 출력: eax += sum
        : "S" (array), "c" (count)   // 입력: rs1=array, rcx=count
        : "memory"
    );

    return sum;
}

int main() {
    int test_array[] = {10, 20, 30, 40, 50};
    int array_count = 5;

    int result_sum = ArraySum_GCC(test_array, array_count);
    printf("Array Sum (GCC style): %d\n", result_sum);

    return 0;
}
```

```
lwy1214@LEGION540-Assembl~:~+~+
[GNOME-nano 6.2] Arraysum.c ~
#include <stdio.h>

int ArraySum_GCC(int array[], int count) {
    int sum = 0;

    __asm__ volatile (
        "movl $0, %%eax\n\t"           // sum = 0
        "test %%rcx, %%rcx\n\t"        // count == 0 ?
        "jne end_loop\n\t"
        "loop_start:\n\t"
        "addl (%rsi), %%eax\n\t"       // eax += *rsi
        "addq $4, %%rsi\n\t"           // rs1 += 4 (int 크기)
        "loop loop_start\n\t"
        "end_loop:\n\t"
        : "+a" (sum)                // 출력: eax += sum
        : "S" (array), "c" (count)   // 입력: rs1=array, rcx=count
        : "memory"
    );

    return sum;
}

int main() {
    int test_array[] = {10, 20, 30, 40, 50};
    int array_count = 5;

    int result_sum = ArraySum_GCC(test_array, array_count);
    printf("Array Sum (GCC style): %d\n", result_sum);

    return 0;
}
```

```
lwy1214@LEGION540-Assembl~:~+~+
[GNOME-nano 6.2] Arraysum.c ~
#include <stdio.h>

int ArraySum_GCC(int array[], int count) {
    int sum = 0;

    __asm__ volatile (
        "movl $0, %%eax\n\t"           // sum = 0
        "test %%rcx, %%rcx\n\t"        // count == 0 ?
        "jne end_loop\n\t"
        "loop_start:\n\t"
        "addl (%rsi), %%eax\n\t"       // eax += *rsi
        "addq $4, %%rsi\n\t"           // rs1 += 4 (int 크기)
        "loop loop_start\n\t"
        "end_loop:\n\t"
        : "+a" (sum)                // 출력: eax += sum
        : "S" (array), "c" (count)   // 입력: rs1=array, rcx=count
        : "memory"
    );

    return sum;
}

int main() {
    int test_array[] = {10, 20, 30, 40, 50};
    int array_count = 5;

    int result_sum = ArraySum_GCC(test_array, array_count);
    printf("Array Sum (GCC style): %d\n", result_sum);

    return 0;
}
```

```
lwy1214@LEGION540-Assembl~:~+~+
[GNOME-nano 6.2] Arraysum.c ~
#include <stdio.h>

int ArraySum_GCC(int array[], int count) {
    int sum = 0;

    __asm__ volatile (
        "movl $0, %%eax\n\t"           // sum = 0
        "test %%rcx, %%rcx\n\t"        // count == 0 ?
        "jne end_loop\n\t"
        "loop_start:\n\t"
        "addl (%rsi), %%eax\n\t"       // eax += *rsi
        "addq $4, %%rsi\n\t"           // rs1 += 4 (int 크기)
        "loop loop_start\n\t"
        "end_loop:\n\t"
        : "+a" (sum)                // 출력: eax += sum
        : "S" (array), "c" (count)   // 입력: rs1=array, rcx=count
        : "memory"
    );

    return sum;
}

int main() {
    int test_array[] = {10, 20, 30, 40, 50};
    int array_count = 5;

    int result_sum = ArraySum_GCC(test_array, array_count);
    printf("Array Sum (GCC style): %d\n", result_sum);

    return 0;
}
```

```
lwy1214@LEGION540-Assembl~:~+~+
[GNOME-nano 6.2] Arraysum.c ~
#include <stdio.h>

int ArraySum_GCC(int array[], int count) {
    int sum = 0;

    __asm__ volatile (
        "movl $0, %%eax\n\t"           // sum = 0
        "test %%rcx, %%rcx\n\t"        // count == 0 ?
        "jne end_loop\n\t"
        "loop_start:\n\t"
        "addl (%rsi), %%eax\n\t"       // eax += *rsi
        "addq $4, %%rsi\n\t"           // rs1 += 4 (int 크기)
        "loop loop_start\n\t"
        "end_loop:\n\t"
        : "+a" (sum)                // 출력: eax += sum
        : "S" (array), "c" (count)   // 입력: rs1=array, rcx=count
        : "memory"
    );

    return sum
```