**REPORT**

**[어셈블리 실습 02]**

원, 폰트, 동전이(가) 표시된 사진

자동 생성된 설명

**과 목 : 시스템소프트웨어**

**담당교수 : 석문기 교수님**

**학 과 : 컴퓨터공학과**

**학 번 : 2021111971**

**이 름 : 이재혁**

텍스트, 클립아트이(가) 표시된 사진

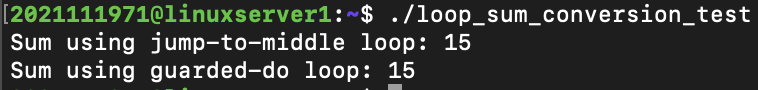
자동 생성된 설명

**1. 어셈블리어 코드**

|  |
| --- |
| **jump\_to\_middle\_loop\_sum.asm** |
| section .text  global jump\_to\_middle\_loop\_sum  jump\_to\_middle\_loop\_sum:  mov edx, 0 ; sum 0으로 초기화  mov eax, 0 ; i 0으로 초기화  jmp .loop\_check ; 처음에 조건 검사 부분으로 점프  .loop\_start:  movsxd rcx, eax ; i 값을 64비트로 부호 확장하여 rcx에 저장  add edx, dword[rdi+rcx\*4] ; sum+=arr[i]  add eax, 1 ; i++  jmp .loop\_check ; 조건 검사 부분으로 점프  .loop\_check:  cmp eax, esi ; i < n 검사  jl .loop\_start ; i < n 이면 .loop\_start로 이동  mov eax, edx ; sum에 저장된 값 eax에 저장  ret ; 함수 반환 |

|  |
| --- |
| **guarded\_do\_loop\_sum.asm** |
| section .text  global guarded\_do\_loop\_sum  guarded\_do\_loop\_sum:  ; todo: 초기화  mov edx, 0 ; sum = 0  mov eax, 0 ; i = 0  ; todo: 초기조건 검사  cmp eax, esi ; i < n  jge .loop\_end ; i >= n 이면.loop\_end로 이동해 반복 종료  .loop\_start:  ; todo: body  movsxd rcx, eax ; i 64비트 확장  add edx, dword[rdi + rcx \* 4] ; sum += arr[i]  add eax, 1 ; i++  ; todo: test  cmp eax, esi ; i < n  jl .loop\_start ; i < n 이면 다시 반복  .loop\_end:  mov eax, edx ; 최종 합계를 eax에 저장  ret; 함수 반환 (eax에 최종 합계가 저장됨) |

**2. 컴파일 후 실행결과**

****

**3. for\_loop\_ctest.s**

for\_loop\_sum

|  |
| --- |
| for\_loop\_sum:  .LFB0:  .cfi\_startproc  endbr64  pushq %rbp  .cfi\_def\_cfa\_offset 16  .cfi\_offset 6, -16  movq %rsp, %rbp  .cfi\_def\_cfa\_register 6  movq %rdi, -24(%rbp) ; arr의 포인터를 rdi 저장  movl %esi, -28(%rbp) ; n의 값을 esi에 저장  movl $0, -8(%rbp) ; sum 0으로 초기화  movl $0, -4(%rbp) ; i를 0으로 초기화  jmp .L2 ; .L2 조건 검사 부분으로 점프  .L3:  movl -4(%rbp), %eax ; i를 eax에 저장  cltq ; eax를 64비트로 확장  leaq 0(,%rax,4), %rdx; i\*4 인덱스 위치를 저장  movq -24(%rbp), %rax ; 배열의 시작주소를 rax에 저장  addq %rdx, %rax ; arr[i]의 주소 계산  movl (%rax), %eax ; arr[i]의 값 eax에 저장  addl %eax, -8(%rbp) ; sum += arr[i]  addl $1, -4(%rbp) ; i++  .L2:  movl -4(%rbp), %eax ; i값 eax에 저장  cmpl -28(%rbp), %eax ; i < n 비교  jl .L3 ; i < n 이면 .L3로 반복  movl -8(%rbp), %eax ; eax에 sum 저장  popq %rbp  .cfi\_def\_cfa 7, 8  ret  .cfi\_endproc |

초기화 이후, 조건 확인 -> jump to middle 변환

**4. dot\_product.asm**

|  |
| --- |
| section .text  global dot\_product  dot\_product:  ; 초기화  mov eax, 0 ; sum = 0  mov ecx, 0 ; i = 0  .loop\_start:  ; i < n 검사  cmp ecx, edx ; i < n 비교  jge .loop\_end ; i >= n 이면 루프 종료  ; vec1[i] \* vec2[i] 계산  movsxd rcx, ecx ; i를64비트 rcx로 확장  mov ebx, dword [rdi + rcx \* 4] ; vec1[i] 값을 ebx에 저장  imul ebx, dword [rsi + rcx \* 4] ; ebx와 vec2[i]곱 ebx에 저장  ; sum += vec1[i] \* vec2[i]  add eax, ebx ; eax에 곱셈 결과를 더함  ; i++  inc ecx ; 인덱스 i를 증가  ; 루프 반복  jmp .loop\_start  .loop\_end:  ; 결과 반환 (eax에 최종 내적 값 저장됨)  ret |

**5. dot\_product\_test.c 테스트**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**