|  |  |
| --- | --- |
|  | |
| 3.60 | |
| 어셈블리 코드 | |
| Loop:  Movl %esi, %ecx;  Movl $1, %edx  Movl $0, %eax  Jmp .L2  .L3:  Movq %rdi, %r8  andq %rdx, %r8  Orq %r8, %rax;  Salq %cl, %rdx  .L2  Testq %rdx, %rdx  Jne .L3  Rep; ret | X =%rdi / n = %esi (=RSI)  %ecx = n  %edx = 1  %eax = 0  L2 로 점프  %r8 = x  %r8 = %r8 & %rdx(=%edx=1)  %rax = %rax or %r8  %rdx = %rdx << %cl(=%ecx = n)  Rdx 가 음수인지 0인지 확인  Rdx 가 0이 아니면 L3으로 점프  반복; 리턴 |
| 1. X, n, result, mask 의 레지스터 | |
| c코드를 확인해보면 for문에서 mask 값에 따라 반복 여부를 결정한다.  즉 %rdx가 mask가 된다. 그러면 자동으로 Loop:에서 0을 넣어준 %eax = %rax가 result가 된다.  답  X = %rdi  N = %esi  Result = %rax  Mask = %rdx | |
| 1. Result와 mask의 초기값은 얼마인가. | |
| Result 의 초기값 = 0  Mask 의 초기값 = 1  by Loop | |
| 1. Mask 의 테스트 조건은 무엇인가. | |
| L2 확인해보면 rdx가 0이 아니면 점프한다.  즉 mask !=0 이다. | |
| 1. 어떻게 mask가 갱신되는가. | |
| Salq 문장을 보면 rdx = rdx <<n 으로 적혀있다.  즉 mask = mask << n 이다. | |
| 1. 어떻게 result가 갱신되는가. | |
| Orq 문장을 보면 rax = rax or r8이다.  여기서 r8은 위에 문장을 통해 x&mask이고 즉 result = result | ( x & mask) 이다.  간단히 쓰면 result |= x&mask | |
| 1. C 코드의 빈칸을 모두 채우시오 | |
| Long loop (long x, long n)  {  Long result = 0;  Long mask;  For ( mask =1; mask!=0; mask = mask << n ) {  Result |= x&mask;  }  Return result  } | |
| 서버 내 c언어 작성 | |
|  | |
| Gcc 컴파일러 이용해 어셈블리어 생성 결과 | |
|  | |

|  |  |
| --- | --- |
| 3.61 | |
| 1. c언어 작성 | |
|  | |
| 1. 어셈블리어 표현 | |
|  | |
| 1. 어셈블리어 표현 확인 | |
| .LFB0  Testq %rdi %rdi  Je .L3  Movq (%rdi), %rax  Ret  .L3  Movl $0, eax  ret | %rdi가 0인지 확인 즉 입력받은 값이 NULL인지 확인  0인경우 (=NULL) L3으로 점프  %rax = 입력받은 포인터값  Rax 리턴  Rax에 0을 넣은후 리턴  즉 movq 인스트럭션 이전에 null값인지 확인후 null값이면 0을 리턴하도록 되어 있어서 널포인터 역참조 에러가 발생하지 않는다. |