컴퓨터 구조 과제 2

컴퓨터 구조 3분반
소프트웨어학과
32170578 김산

<문제>

Linux 시스템에서 nxn 행렬의 곱셈을 구현하는 함수 matmul()를 구현하고 실행 결과를 제시하라. 또한 -O1 옵션을 사용하여 gcc로 컴파일한 후 생성된 코드를 분석하고 코드 내에서 사용된 각 register의 용도를 설명하라.

<코드>

C code

```c
#include <stdio.h>
#define N 3 //NxN행렬
int * matmul(int * mat1,int * mat2,int * matres){
    int I, j, k; //for loop index
    int sum;
    /*행렬 곱 수행*/
    for (int i =0; i < N; i ++)
    {
        for (int j =0; j < N; j ++)
        {
            sum =0;
            for (int k =0; k < N; k ++)
            {
                sum += mat1[i *N +k] * mat2[k *N +j];
            }
            matres[i*N +j] = sum;
        }
    }
}
int main(){
    int matres[N *N];
    int mat1[N *N] = {1,2,3,1,2,3,1,2,3};
    int mat2[N *N] = {1,2,3,1,2,3,1,2,3};
    matmul(mat1,mat2,matres);
    for (int i =0; i < N; i ++)
    {
        for (int j =0; j < N; j ++)
        {
            printf("%d ", matres[i*N+j]);
        }
        printf("₩n");
    }
    return 0;
}
```

<실행결과>

{1,2,3,1,2,3,1,2,3} x {1,2,3,1,2,3,1,2,3} (3x3행렬)

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

6 12 18
6 12 18
6 12 18
[1] + Done                       "/usr/bin/gdb" --i
.6qy"
user32170578@oss-contribute:~$
```

<실행파일 분석>

matmul함수 objdump -d 결과

```
00000000000006f5 <matmul>:
 6f5:   41 bb 00 00 00 00       mov    $0x0,%r11d
 6fb:   eb 49                   jmp    746 <matmul+0x51>
 6fd:   46 89 0c 92             mov    %r9d,(%rdx,%r10,4)
 701:   49 83 c2 01             add    $0x1,%r10
 705:   49 83 fa 03             cmp    $0x3,%r10
 709:   74 29                   je     734 <matmul+0x3f>
 70b:   4e 8d 04 96             lea    (%rsi,%r10,4),%r8
 70f:   b8 00 00 00 00          mov    $0x0,%eax
 714:   41 b9 00 00 00 00       mov    $0x0,%r9d
 71a:   8b 0c 87                mov    (%rdi,%rax,4),%ecx
 71d:   41 0f af 08             imul   (%r8),%ecx
 721:   41 01 c9                add    %ecx,%r9d
 724:   48 83 c0 01             add    $0x1,%rax
 728:   49 83 c0 0c             add    $0xc,%r8
 72c:   48 83 f8 03             cmp    $0x3,%rax
 730:   75 e8                   jne    71a <matmul+0x25>
 732:   eb c9                   jmp    6fd <matmul+0x8>
 734:   41 83 c3 03             add    $0x3,%r11d
 738:   48 83 c2 0c             add    $0xc,%rdx
 73c:   48 83 c7 0c             add    $0xc,%rdi
 740:   41 83 fb 09             cmp    $0x9,%r11d
 744:   74 08                   je     74e <matmul+0x59>
 746:   41 ba 00 00 00 00       mov    $0x0,%r10d
 74c:   eb bd                   jmp    70b <matmul+0x16>
 74e:   c3                      retq
```

코드		용도
<matmul>		
mov	$0x0,%r11d	r11d : i의 값(0으로 초기화)
jmp	746 <matmul+0x51>	<matmul+0x51>로 이동
<matmul+0x8>		
mov	%r9d,(%rdx,%r10,4)	r9d : sum의 값 rdx : matres배열의 주소값 r10 : j의 값
add	$0x1,%r10	r10 : j의 값(1을 더함)
cmp	$0x3,%r10	r10 : j의 값(N(3)과 비교)
je	734 <matmul+0x3f>	비교값이 같으면 <matmul+0x3f>로 이동
<matmul+0x16>		
lea	(%rsi,%r10,4),%r8	rsi : 인자에서 전달받은 mat2 배열의 주소값 r10 : j의 값(mat2배열의 인덱스) r8 : mat2[i*N+k]의 주소값을 전달받음
mov	$0x0,%eax	eax : k의 값(0으로 초기화)
mov	$0x0,%r9d	r9d : sum의 값(0으로 초기화)
<matmul+0x25>		
mov	(%rdi,%rax,4),%ecx	rdi : 인자에서 전달받은 mat1 배열의 주소값 rax : k의 값(mat1배열의 인덱스) ecx : mat1[k*N+j] 값을 저장
imul	(%r8),%ecx	r8 : mat2[i*N+k] 값 ecx : mat2[i*N+k] 와 mat1[k*N+j]의 값을 곱한 값 저장
add	%ecx,%r9d	r9d : sum의 값(sum += mat2[i*N+k] * mat1[k*N+j])
add	$0x1,%rax	rax : k의 값(1을 더함)
add	$0xc,%r8	r8 : mat2의 주소값(12(sizeof(int) * 3)을 더하여 저장)
cmp	$0x3,%rax	rax : k의 값(N(3)과 비교한다)
jne	71a <matmul+0x25>	비교값이 작으면 <matmul+0x25>로 이동
jmp	6fd <matmul+0x8>	<matmul+0x8>로 이동
<matmul+0x3f>		
add	$0x3,%r11d	r11d: i의 값(3을 더한다)
add	$0xc,%rdx	rdx : matres배열의 주소값(12(N*sizeof(int))를 더하여 저장)
add	$0xc,%rdi	rdi : mat1배열의 주소값(12(N*sizeof(int))를 더하여 저장)
cmp	$0x9,%r11d	r11d: i의 값(12와 비교)
je	74e <matmul+0x59>	비교값이 같으면 <matmul+0x59>로 이동
<matmul+0x51>		
mov	$0x0,%r10d	r10d : j의 값(0으로 초기화)
jmp	70b <matmul+0x16>	70b <matmul+0x16>로 이동
retq		결과값 반환