**알고리즘 과정 분석 리포트**

**레지스터**

레지스터의 공간은 16개이고 행렬로 계산되어야 할 숫자는 18개이기 때문에 행렬1,2 중에서 행렬2에게 레지스터 자리 9개를 고정적으로 제공했습니다. 나머지 레지스터의 6개 공간 중 3개는 행렬1에서 각 행마다의 값들로 초기화했습니다. 남은 레지스터 3개의 공간은 사용하지 않았습니다.

예를 들어, C0, C1, C2 값을 계산하기 위해서는 행렬 1의 첫번째 행이 필요하므로 레지스터 R0, R1, R2에 그 값들로 초기화했고 행렬을 계산하는 과정에서 나오는 값들 또한 다시 R0, R1, R2에 덮어씌워 마지막 결과값을 R0에 오도록 설계했습니다. 자세하게 말하자면, C(0~8)의 값들 중 하나의 값을 구하는 과정에서 MUL 명령어의 결과값들은 각각 R(0~2)에 덮어씌워지고 이 값들이 ADD 명령어를 거치면 항상 R0에 값이 저장되어 최종적인 결과값은 R0에 오게 됩니다.

이렇게 행렬의 결과값을 계산하기 위해서는 C(0~2)가 계산되고 나서 다시 R(0~2)를 행렬 1의 두 번째 행의 값들로 초기화 되어야 하고 마찬가지로 C(3~5)가 계산되고 나서는 R(0~2)가 다시 행렬 1의 세번째 행의 값들로 초기화 되어야 합니다.

이를 토대로 레지스터를 분석하자면 R(0~2)는 그 속의 데이터가 계속 변화하고, R(3~11)은 값이 고정 되어있고, R(12~15)는 값 자체가 입력되지 않은 상태입니다. 계산을 다 마치고 나서는 R(0~2)의 값을 0으로 초기화했습니다.

**레지스터와 메모리**

레지스터 R(0~11)의 데이터를 MOV3 명령어를 이용하여 초기화할 때마다 바로 MOV1 명령어를 이용하여 메모리에 그 값을 저장했습니다. R(3~11)의 데이터는 고정적이므로 처음에 메모리 M[9~17]의 자리에 값을 저장했습니다. R(0~2)의 데이터는 행렬1,2를 계산한 결과로 나오는 행렬 3의 어떤 행을 계산하는가에 따라서 값이 변화하기 때문에 처음 R(0~2)의 데이터는 M[0~2], 두 번째 행을 계산할 때의 R(0~2)의 데이터는 M[3~5], 세번째 행을 계산할 때의 R(0~2)의 데이터는 M[6~8]에 순차적으로 저장했습니다. 행렬3의 데이터는 그 값이 나올 때마다 바로 메모리 M[18~26]에 저장했습니다. 이렇게 되면 값이 변화하는 레지스터는 R(0~2) 뿐이므로 메모리로부터 데이터를 받아오는 레지스터 또한 R(0~2) 뿐입니다.

이러한 과정을 거쳐 메모리 M[0~8]에는 행렬 1의 값들이, M[9~17]에는 행렬 2의 값들이, M[18~26]에는 행렬 3의 값들이 저장되어집니다.