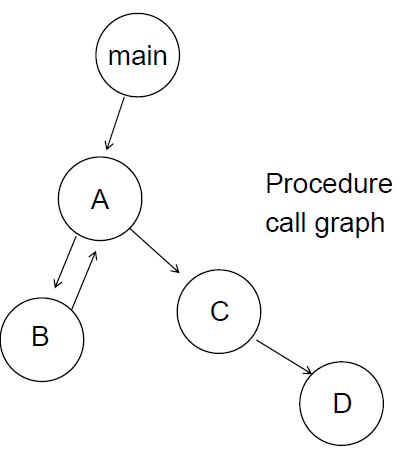
#HW9 2017069598 박상지

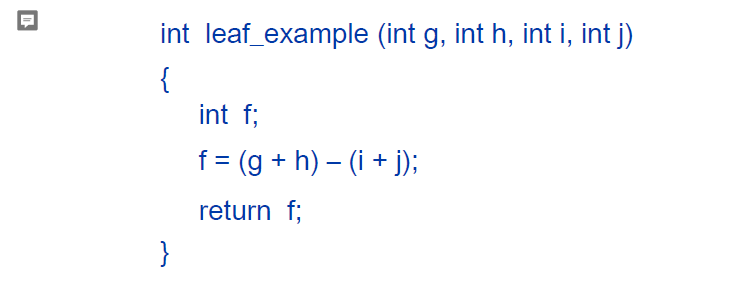
run c programs, procedure calls

프로그램이 실행시킨다는 것은?



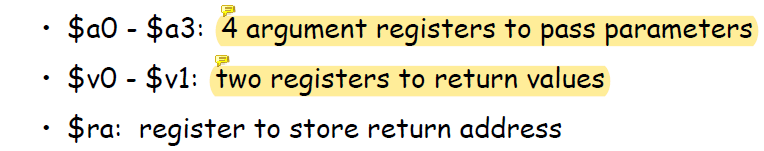
프로그램을 실행시킨다는 것은 ISA 관점에서 machine instruction을 하나씩 실행시킨 것, 또는 fetch-decode-execute을 반복적으로 실행하는 것이다. HLL(high level language) 관점에서는 statement를 하나씩 실행시키는 것, 또는 function call return이 반복되는 것이다(procedure level).

leaf procedure



해당 코드에서는 function에 대한 argument가 4개가 있는 간단한 코드가 있다. leaf procedure은 call graph에서 맨 끝에 위치해 있는 procedure, 즉 다른 procedure을 call하지 않는 것이다.

procedure에서는 caller와 callee로 나누어 지는데 caller 관점에서는 다른 callee를 call 하기 때문에 jump and link로 return address를 저장해야 한다. callee 입장에서는 caller가 사용한 레지스터 값을 생각을 해서 기존의 있는 레지스터 값을 백업을 해서 마음데로 쓸 수 있게 원하는 계산을 하고 적절한 방법으로 결과를 jump register를 사용하여 caller로 return 한다.



32개의 레지스터에서 4개의 argument 전용 레지스터를 따로 만들고, 2개의 return value 레지스터를 지정하였다. ra는 return address를 저장하는 레지스터이다.

질문) leaf procedures에서 4개 이상의 argument보다 필요한다면?

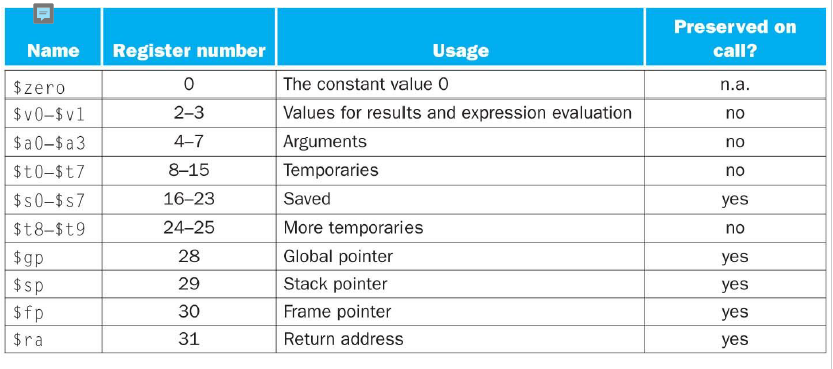
벤치 마크 프로그램 조사 결과 통상 argument의 사용 개수가 4개 이하였다. 그렇기 때문에 argument register를 4개로 지정한 것이다. 하지만 그 4개 수보다 많이 필요한다면 메모리에 접근하여 값을 할당하여 사용해야한다. 이럴 경우 속도는 줄어 든다.

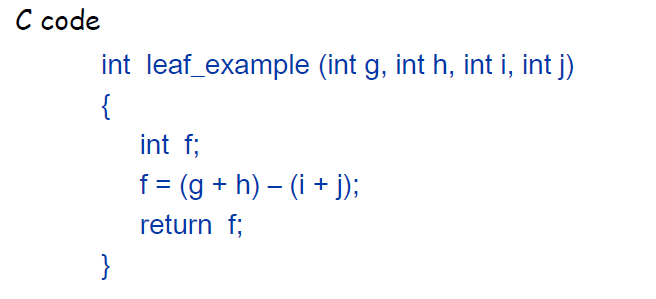
질문) return value의 register은 왜 2개 필요한가?

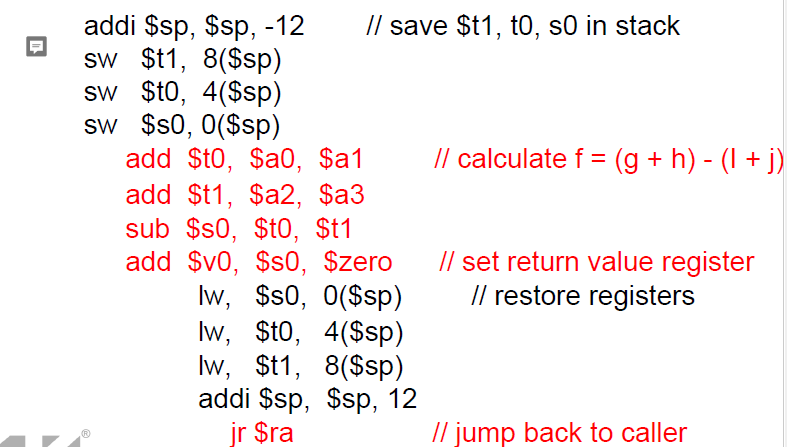
c프로그램은 return 값을 1개만 지정한다. 하지만 double return value는 64-bit의 공간이 필요하기 때문에 double value를 return 할 때를 대비하여 레지스터 2개를 만들어 놓은 것이다.

질문) 하나 이상의 아이템을 return할 때에는 어떻게 하는가?

여러 개의 아이템을 return을 할 때에는 구조체로 묶어서 포인터로 주소 값을 return을 하든가, executable value을 사용하여 return을 한다.



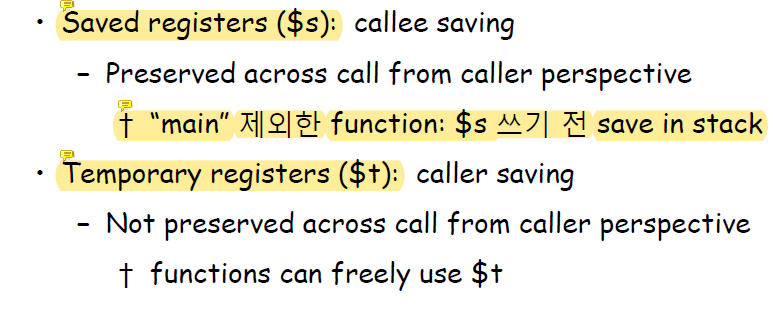




해당 C코드에 대한 MIPS 코드이다. 빨강색으로 칠한 부분이 실제로 코드가 실행되는 부분이다. 만약 해당 C코드를 실행하려면 s0, t0, t1 레지스터를 사용한다는 가정하에 해당 3개의 레지스터가 caller에서 중요한 값이 저장될 수 있기 때문에 stack 레지스터에 복원을 해야 한다. 3개 이기 때문에 12의 주소 값을 할당해줘야 하고, 실제 실행 코드를 실행시킨 다음에 다시 스택 레지스터에 저장된 값을 백업을 해줘야 한다. 그 다음에 return을 해준다.

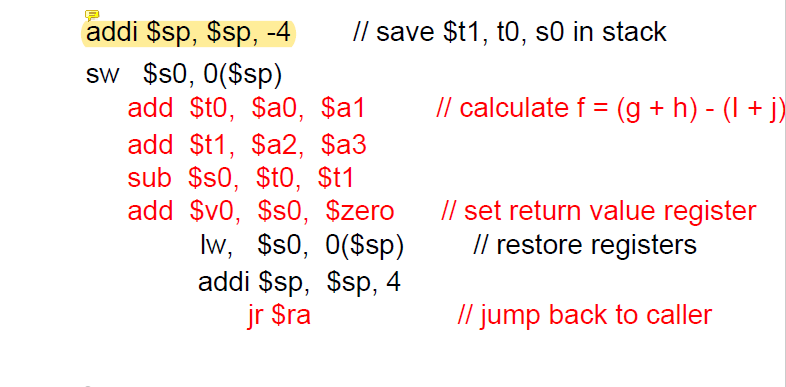
-컴파일러는 해당 레지스터에 값이 들어 있는지 알 수 없다. 최악의 상황을 생각해야 하기 때문에 안전하게 callee는 스택 레지스터에 복원 값을 저장해야 한다.

coordinating register usage

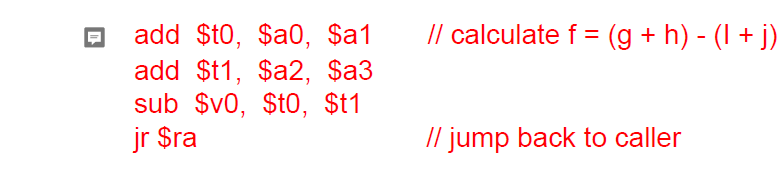


saved register는 callee 입장에서 복원을 담당하는 부분이기 때문에 caller 입장에서는 따로 백업을 할 필요가 없다. callee에서 function return call해서 돌아와도 해당 레지스터는 백업이 돼서 돌아온다. (이미 callee에서 saving을 해서 복원을 했기 때문이다.)

caller 입장에서 다른 function을 부르고 돌아올 경우 임시 레지스터는 바꿔서 돌아올 가능성이 있기 때문에 caller에서 백업을 해야 한다.



t 레지스터는 백업을 할 필요가 없기 때문에 해당 s 레지스터만 백업을 한다. 위에 MIPS 코드를 더욱 간단하게 한 것이다.



처음부터 saved register을 사용하지 않고 임시 레지스터의 연산 값을 value register에 저장해서 return 한다.