**Test #10. 2012.5.17 프로그래밍언어론 01 분반- 개인 문제**

학번:

이름:

조번호:

1. 다음 각 항목이 참이면 O, 아니면 X 를 하시오.

* 1. Out mode와 Inout mode의 parameter 값은 subprocedure call 시점에서 actual parameter에 전달된다.
  2. Recursive call 이 있는 함수는 결과값을 return해야 하므로 ARI에 결과값을 위한 장소가 필요하다.
  3. 다음 함수가 호출되어 수행되는 동안 (ㄱ) 시점에서, 변수 x는 ARI내의 parameter용 공간에 위치하고, 변수 j는 ARI 내에 local variable 용 공간에 위치한다.

int inc(int x){

static int j=0;

j+= x; //(ㄱ)

return j;

}

☞ Static scoping rule 을 따르는 비지역변수를 찾는 방법에서

* 1. Static parent 의 가장 최근 ARI의 바닥 주소를 찾는 방법 중 하나는 dynamic chain 을 따라가며 처음으로 만나는 static parent의 ARI를 찾는 방법이 있다.
  2. ~~callee의 static parent는 caller의 static ancestor 중 하나이다.~~ (O를 의도했지만, 자기자신은 ancestor에 안 들어가므로 모호하다는 주장으로 채점 안함.)
  3. chain offset이란 변수를 참조한 block의 static depth와 참조한 변수를 선언한 block의 static depth의 차 (difference)이다.
  4. static chain을 사용하여 static scoping rule을 따르는 변수들을 접근하는 경우 호출 depth가 늘어나면 시간 비용이 증가한다.
  5. 실행중인 subprogram의 static depth가 n인 경우 display에는 n+1개의 값이 유효하다. (단 depth는 0부터 시작이다.)
  6. ARI의 꼭대기에 block 지역변수가 사용할 기억장소를 할당하는 방법을 쓴다면, 다음 subprogram에서 ARI내에 지역변수가 사용할 기억장소 크기를 6\*4byte 로 하면 낭비도 없고 부족함도 없이 적당하다.

sub(){

int x;

while(1) { int a; int b;}

{int c; …}

{ind d; …}

{int e; …}

}

* 1. 다음 코드에서 g()가 h()를 호출 할 때, display 관리를 위한 아래 i~v 스텝 중에서 ii와 v 중 적어도 하나는 불필요한 과정이다.

|  |  |
| --- | --- |
| main(){  h(){…}  g(){ … h(); …}    g();  } | * + 1. P의 ARI(P-ARI)를 생성     2. P-ARI 안에 display[k]를 저장     3. Display[k]가 P-ARI를 point하도록 함     4. Subprogram P 실행     5. P-ARI에 저장해둔 이전의 display[k] 값을 display[k]에 저장 |

1. 다음 그림은 Sub1, Sub2, Sub3, Bigsub의 subprogram들과 Main으로 이루어진 프로그램을 수행했을 때 구성된 runtime stack의 이다. 본 그림으로부터 유추할 수 있는 subprogram들의 nesting 구조와 각 변수가 선언된 위치 (어느 subprogram의 내부 인지) 를 파악하여 원래 프로그램을 복원해보시오. 단 실행 코드 부분은 복원하지 않는다.

**Test #10. 2012.5.17 프로그래밍언어론 01 분반 조별 문제**

조번호:

참여자 (학번, 이름)

-

-

-

-

-

1. 연상

2. 다음 프로그램이 dynamic access rule의 shallow access 방식으로 접근된다면, main으로 부터 시작하여 함수 sub2()이 처음으로 호출되었을 때 symbol table의 모습을 그려보시오.

void sub3() { int x, z; sub2();...}

void sub2() { int x; x = y + w + z; ......}

void sub1() { int z, w; ... sub3() ...; }

void main() { int x, y; ... sub1(); ... }