**Test #10. 2013.5.13 프로그래밍언어론 01 분반**

학번:

이름:

1. 다음 각 항목이 참이면 O, 아니면 X 를 하시오.
2. C는 ARI내에 static link 가 필요하지 않다.
3. Dynamic scoping rule을 따르는 언어에서, 현재 수행중인 함수가 sub3이고 변수 u, z, w를 사용하려 한다고 가정한다면, 다음 shallow access 테이블에서 main에 정의된 u와 sub3에 정의된 z와 sub2에 정의된 w를 사용한다.

u

v

x

z

w

main

main

sub2

sub3

sub2

sub1

sub3

sub1

1. Static chain을 사용하여 static scoping rule을 따르는 변수들을 접근하는 경우 호출, depth가 늘어나면 수행 시간이 증가한다.
2. Static scoping rule 을 따르는 비지역변수를 찾는 방법에서, 변수가 포함된 ARI 안에서 변수가 EP(환경포인터)로부터 떨어져 있는 거리를 chain offset이라고 한다.
3. Block 지역변수가 사용할 기억장소를 ARI의 꼭대기에 할당하는 방법에서, 다음 subprogram에서 Block 지역변수가 사용할 기억장소의 최대 크기는 5\*4byte이다.

sub(){

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

{int c; …}

{ind d; …}

{int e; …}

1. AR (activation record) 에 들어가는 항목을 4개이상 나열하시오.
2. 다음 코드에서 g()에서 h()를 호출 할 때, display 관리를 위한 아래 i~v 스텝 중에서 불필요한 과정(들)을 고르고 이유를 간단히 적으시오. (단, main, g, h의 depth는 각각 1, 2, 3라 하자)

main() {

g() {

h(){…/\* no subprogram definitions or calls \*/ }

**call h();**

}

call g();

}

1. h의 ARI(h-ARI)를 생성
2. h-ARI 안에 display[3]를 저장
3. display[3]가 h-ARI를 point하도록 함
4. Subprogram h 실행
5. h-ARI에 저장해둔 이전의 display[3] 값을 display[3]에 저장

답:

1. 다음 각 항목이 참이면 O, 아니면 X 를 하시오.
2. C는 ARI내에 static link 가 필요하지 않다. O
3. Dynamic scoping rule을 따르는 언어에서, 현재 수행중인 함수가 sub3이고 변수 u, z, w를 사용하려 한다고 가정한다면, 다음 shallow access 테이블에서 main에 정의된 u와 sub3에 정의된 z와 sub2에 정의된 w를 사용한다. X

u

v

x

z

w

main

main

sub2

sub3

sub2

sub1

sub3

sub1

1. Static chain을 사용하여 static scoping rule을 따르는 변수들을 접근하는 경우 호출, depth가 늘어나면 수행 시간이 증가한다. X
2. Static scoping rule 을 따르는 비지역변수를 찾는 방법에서, 변수가 포함된 ARI 안에서 변수가 EP(환경포인터)로부터 떨어져 있는 거리를 chain offset이라고 한다. X
3. Block 지역변수가 사용할 기억장소를 ARI의 꼭대기에 할당하는 방법에서, 다음 subprogram에서 Block 지역변수가 사용할 기억장소의 최대 크기는 5\*4byte이다. X

sub(){

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

{int c; …}

{int d; …}

{int e; …}

1. AR (activation record) 에 들어가는 항목을 4개이상 나열하시오.

local variables, parameter, dynamic link, return address, static link

1. 다음 코드에서 g()에서 h()를 호출 할 때, display 관리를 위한 아래 i~v 스텝 중에서 불필요한 과정(들)을 고르고 이유를 간단히 적으시오. (단, main, g, h의 depth는 각각 1, 2, 3라 하자)

답) ii, 이유는 depth 3에 있는 함수는 h 하나이므로 굳이 기존의 display[3]를 저장할 필요가 없다.

main() {

g() {

h(){…/\* no subprogram definitions or calls \*/ }

**call h();**

}

call g();

}

1. h의 ARI(h-ARI)를 생성
2. h-ARI 안에 display[3]를 저장
3. display[3]가 h-ARI를 point하도록 함
4. Subprogram h 실행
5. h-ARI에 저장해둔 이전의 display[3] 값을 display[3]에 저장