**Test #10. 2013.5.14 프로그래밍언어론 02 분반**

학번:

이름:

1. 다음 각 항목이 참이면 O, 아니면 X 를 하시오.
2. Subprogram에서 ARI는 실행 과정에서 변하지 않는 code 부분을 나타낸다.
3. Scope nesting 이 없다면 static link는 AR에 있을 필요가 없다.
4. Static scoping rule 을 따르는 경우, callee의 static parent는 caller의 static ancestor 거나 caller 자신이다.
5. Static scoping rule 을 따르는 비지역변수를 찾는 방법에서, 변수가 포함된 ARI 안에서 변수가 EP(환경포인터)로부터 떨어져 있는 거리를 local offset이라고 한다.
6. 다음 코드에서 g()가 h()를 호출 할 때, display 관리를 위한 아래 i~v 스텝 중에서 ii는 안 해도 괜찮다. (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
6. Block 지역변수가 사용할 기억장소를 ARI의 꼭대기에 할당하는 방법에서, 다음 subprogram에서 Block 지역변수를 위해 추가로 최소 몇바이트를 할당할 필요가 있는가? 단 int 변수 하나는 4byte이다. ( ) byte

sub(){

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

{int c; …}

{ind d; …}

{int e; …}

1. 다음 프로그램이 dynamic scoping rule을 가지고 수행될 때 sub가 호출되어 수행 중 시점의 shallow access table을 그리시오.

void sub() {

int v, w; .....

}

void main() {

int u, v; ... sub(); ...

}

답:

1. 다음 각 항목이 참이면 O, 아니면 X 를 하시오.
2. Subprogram에서 ARI는 실행 과정에서 변하지 않는 code 부분을 나타낸다. X
3. Scope nesting 이 없다면 static link는 AR에 불필요한 정보이다. O
4. Static scoping rule 을 따르는 경우, callee의 static parent는 caller의 static ancestor 거나 caller 자신이다. O
5. Static scoping rule 을 따르는 비지역변수를 찾는 방법에서, 변수가 포함된 ARI 안에서 변수가 EP(환경포인터)로부터 떨어져 있는 거리를 local offset이라고 한다. O
6. 다음 코드에서 g()가 h()를 호출 할 때, display 관리를 위한 아래 i~v 스텝 중에서 ii는 안 해도 괜찮다. (단, main, g, h의 depth는 각각 1, 2, 3라 하자) O

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
6. Block 지역변수가 사용할 기억장소를 ARI의 꼭대기에 할당하는 방법에서, 다음 subprogram에서 Block 지역변수를 위해 추가로 최소 몇바이트를 할당할 필요가 있는가? 단 int 변수 하나는 4byte이다. ( 2\*4 ) byte

sub(){

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

{int c; …}

{ind d; …}

{int e; …}

1. 다음 프로그램이 dynamic scoping rule을 가지고 수행될 때 sub가 호출되어 수행 중 시점의 shallow access table을 그리시오.

void sub() {

int v, w; ...

}

void main() {

int u, v; ... sub(); ...

}

s

m m s

u v w