Activation Records

CS132

March 7, 2011

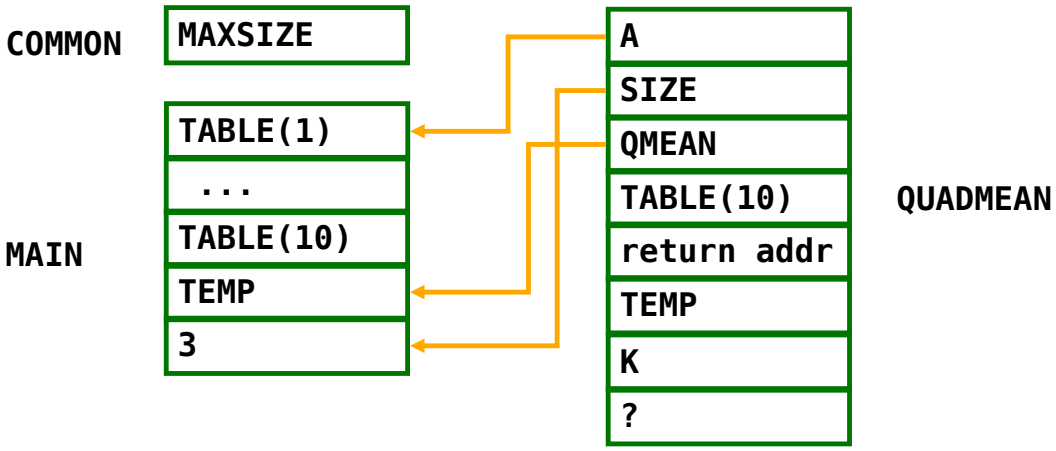
CS132

Static Allocation

```
      PROGRAM TEST
       COMMON MAXSIZE
       REAL TABLE(10), TEMP
       MAXSIZE = 10
       READ *, TABLE(1), TABLE(2), TABLE(3)
       CALL QUADMEAN(TABLE, 3, TEMP)
       PRINT *, TEMP
      END

      SUBROUTINE QUADMEAN(A,SIZE,QMEAN)
       COMMON MSIZE
       INTEGER SIZE, K
       REAL A(SIZE), QMEAN, TEMP
       TEMP = 0.0
       IF ((SIZE.GT.MSIZE).OR.(SIZE.LT.1)) GOTO 99
       DO 10 K = 1, SIZE
         TEMP = TEMP + A(K) * A(K)
10     CONTINUE
99     QMEAN = SQRT(TEMP/SIZE)
       RETURN
      END
```

CS132

Memory Layout



CS132

Typical Fortran Style

```
SUBROUTINE SSYEVD(JOBZ, UPLO, N, A, LDA, W, WORK, LWORK, IWORK, LIWORK, INFO )

*  .. Scalar Arguments ..
   CHARACTER JOBZ, UPLO
   INTEGER INFO, LDA, LIWORK, LWORK, N

*  .. Array Arguments ..
   INTEGER IWORK( * ) REAL A( LDA, * ), W( * ), WORK( * )

*  A      (input/output) REAL array, dimension (LDA, N)
*         On entry, the symmetric matrix A.
*  WORK   (workspace/output) REAL array, dimension (LWORK)
*         On exit, if INFO = 0, WORK(1) returns the optimal LWORK.
*
*  LWORK  (input) INTEGER
*         The dimension of the array WORK.
*         If N <= 1,            LWORK must be at least 1.
*         If JOBZ = 'N' and N > 1, LWORK must be at least 2*N+1.
*         If JOBZ = 'V' and N > 1, LWORK must be at least
*                                  1 + 6*N + 2*N**2.
```

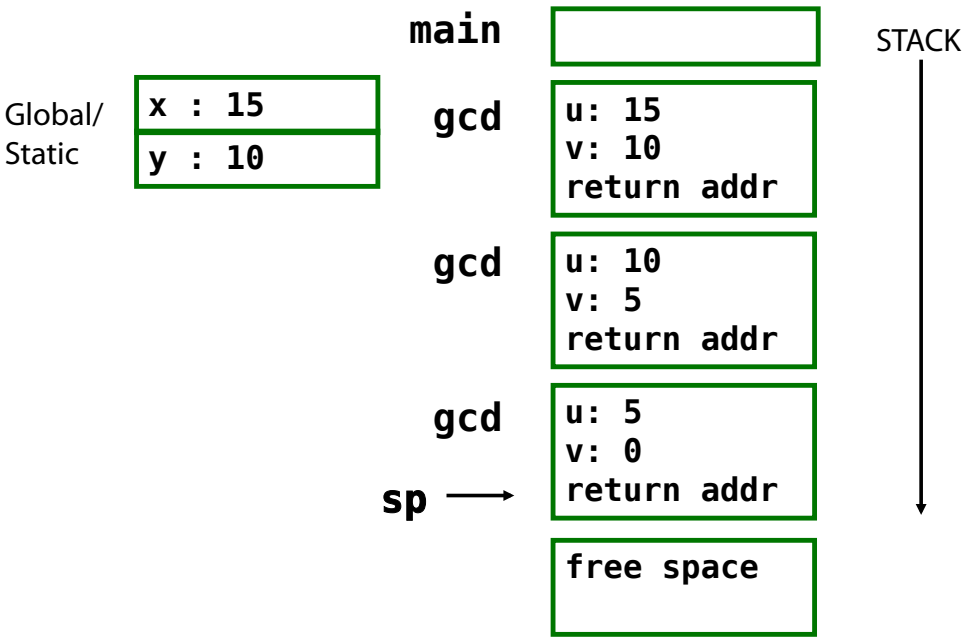
CS132

Stack Allocation

```
#include <stdio.h>

int x,y;

int gcd(int u, int v) {
  if (v == 0)
    return u;
  else
    return gcd(v, u % v);
}

int main() {
  scanf("%d%d", &x, &y);
  printf("%d\n", gcd(x,y));
  return 0;
}
```

CS132

Possible Memory Layout



CS132

Stack Allocation

```
int g(int n) {
  int a[n++];
  for (int i = 0; i < n; ++i)
     a[i] = i;
  int sum = 0;
  for (int i = 0; i < n; ++i)
     sum += a[i];
  return sum;
}

int main() {
  int m;
  scanf("%d", &m);
  printf("%d\n", sum(m));
  return 0;
}
```

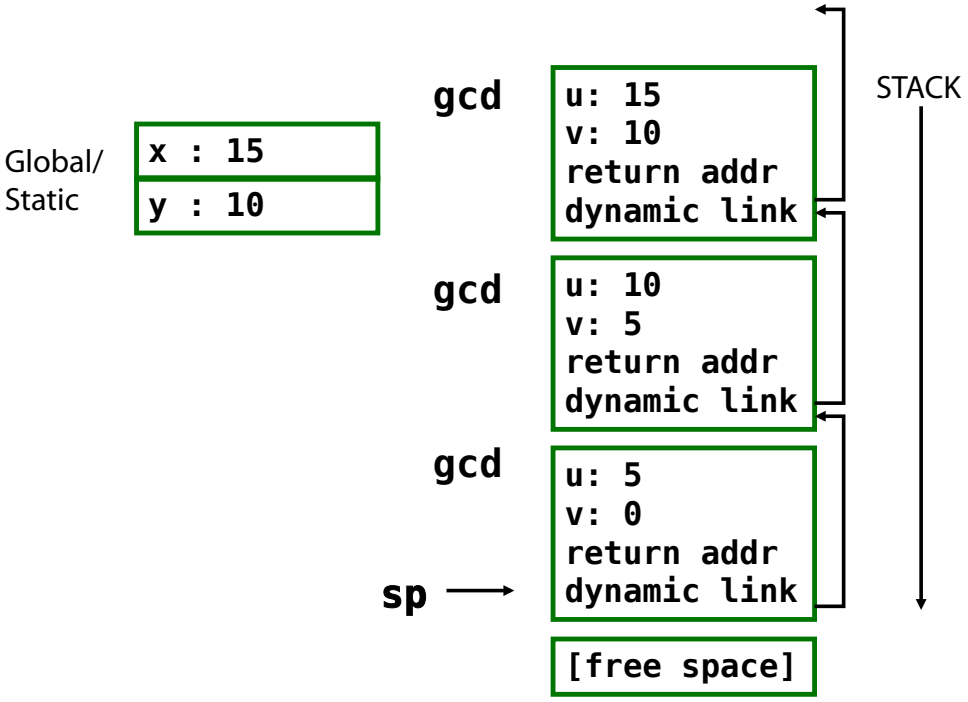
CS132

Stack Allocation

```
int open2 (char *str1, char *str2, int flags, int mode)
{
  char *name =
    (char *) alloca (strlen(str1) + strlen(str2) + 1);
  strcpy (name, str1);
  strcat (name, str2);
  return open (name, flags, mode);
}
```

CS132

Memory Layout



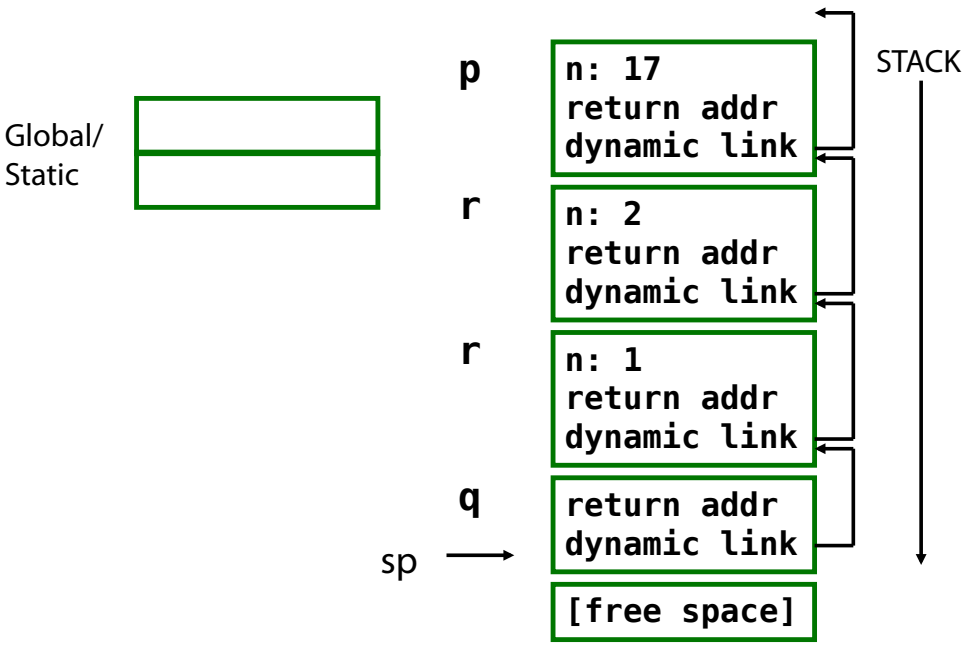
CS132

Nested Functions

```
procedure p;
    var n: integer;

    procedure q;
    begin
       n := n+1;
    end;

    procedure r(n:integer);
    begin
       if n>1 then r(n-1); else q;
    end;

begin
  n := 17;
  r(2);
end;
```

CS132

Memory Layout?



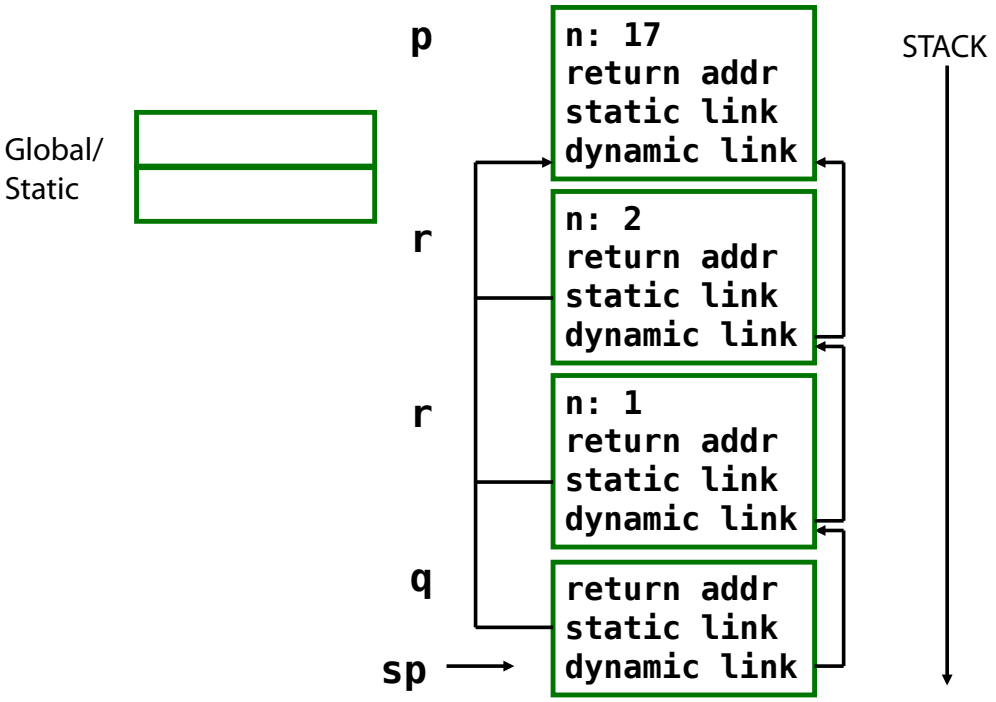
CS132

Does This Work?

At compile time, can we generate code to find each variable?

CS132

Option 1: Static Links



CS132

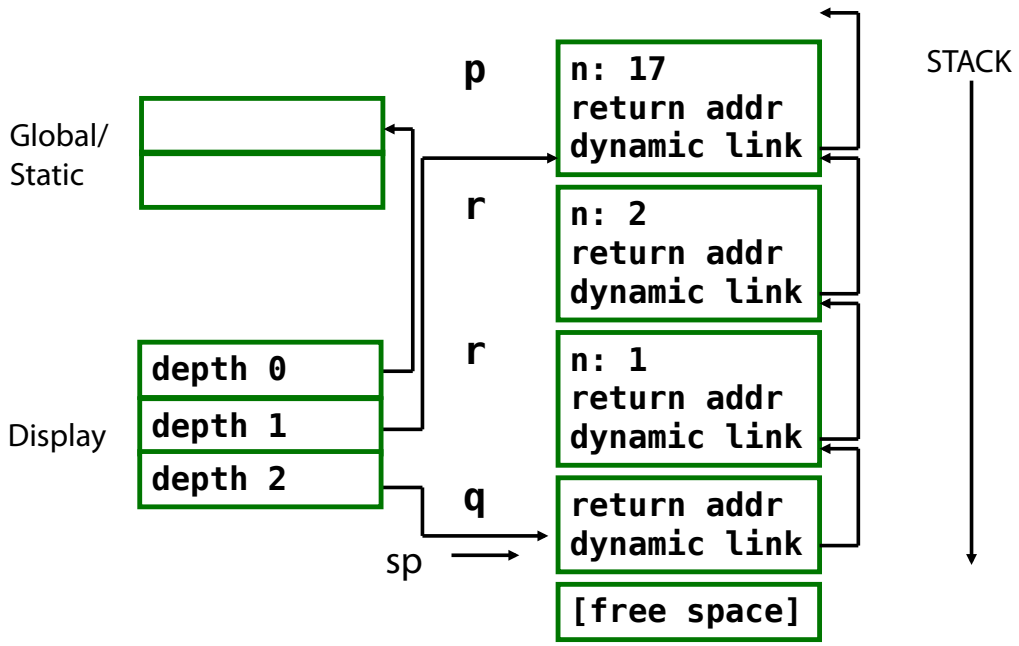
Does This Work?

At compile time, can we generate code to find each variable?

How can we figure out what the static link should be when each frame is created?

CS132

Option 2: Displays



CS132

Does This Work?

At compile time, can we generate code to find each variable?

How can we figure out what the static link should be?

CS132

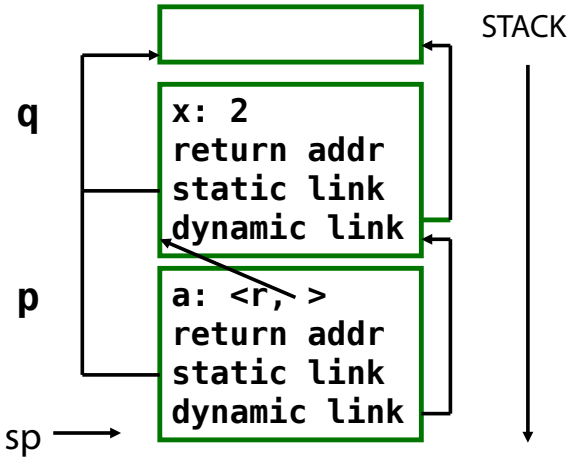
Procedures as Parameters

```
procedure p(procedure a);
  begin
     a
  end;

procedure q;
  var x :integer;

  procedure r;
    var z : int
    begin
      z := 1;
      writeln(x+z)
    end;

  begin
    x := 2;
    p(r)
  end;
```

How does **r** find **x**?

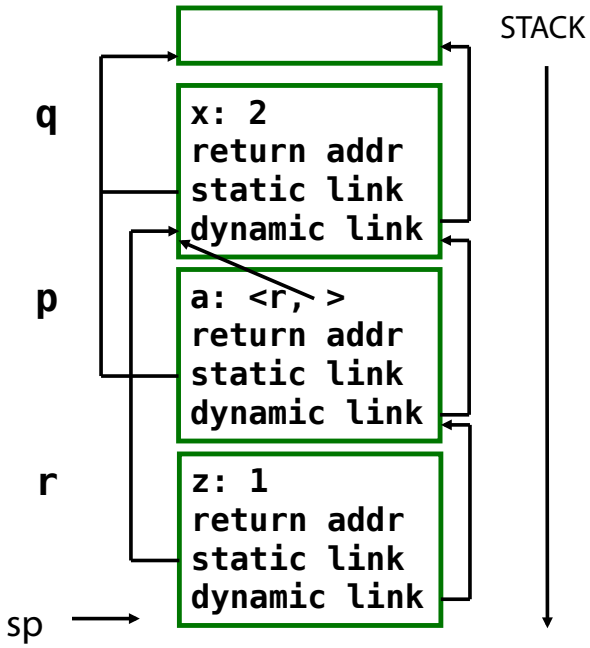
CS132

Before Call to **a**



CS132

During Call to **a**



CS132

Procedures as *Results*

```
let
   add x =
     let
        s = "abc...z"
        f y = x+y
     in
        f
   id = add 0
   g() = id 2
in
   g()
```

CS132

Before **`add 0`** Returns