|  |  |
| --- | --- |
| C Code | Assembly Code |
| int foo();  int main(void) {  int y;  y = foo(); // y = 6  y = foo(); // y = 6  y = foo(); // y = 6  while(1);  }  int foo() {  int x = 5; // x is a local variable  x = x + 1;  return(x)  } | AREA static\_demo, CODE  EXPORT \_\_main  ALIGN  ENTRY  \_\_main PROC  BL foo ; r0 = 6  BL foo ; r0 = 6  BL foo ; r0 = 6  stop B stop  ENDP    foo PROC  **MOV r0,#5**  **ADD r0,r0,#1**  **BX lr**  ENDP  END |

|  |  |
| --- | --- |
| C Code | Assembly Code |
| int foo();  int main(void) {  int y;  y = foo(); // y = 6  y = foo(); // y = 7  y = foo(); // y = 8  while(1);  }  int foo() {  // x is initialized only once  **static** int x = 5; // a static variable  x = x + 1;  return(x)  } | AREA static\_demo, CODE  EXPORT \_\_main  ALIGN  ENTRY  \_\_main PROC  BL foo ; r0 = 5  BL foo ; r0 = 6  BL foo ; r0 = 7  stop B stop  ENDP    foo PROC  **LDR r1,=x**  **LDR r0,[r1]**  **ADD r0,r0,#1**  **STR r0,[r1]**  **BX lr**  ENDP  AREA myData, DATA  ALIGN  **x DCD 5**  END |

|  |  |
| --- | --- |
| C Program | Assembly Program |
| int isPal(int);  int main(){  int n;  n = isPal(12321);  while(1);  }  // Check palindrome number  int isPal(int n){  **static int sum = 0;**  int r;  if(n!=0) {  r = n % 10;  sum = sum\*10 + r;  isPal (n/10);  }    if (sum == n)  return 1;  else  return 0;  } | AREA palindrome, CODE  EXPORT \_\_main  ALIGN  ENTRY  \_\_main PROC  LDR r0,=12321  BL isPal  stop B stop  ENDP  ; Recursively check palindrome  isPal PROC  PUSH {r4,lr}  MOV r4,r0  CBZ r4,done ; if n is zero, done  MOV r2,#10  SDIV r1,r4,r2 ; r1 = n/10  MLS r3,r1,r2,r4 ; r3 = r = n - r1 \* 10;  **LDR r1,=sum**  **LDR r1,[r1] ; r1 = sum**  ADD r1,r1,r1,LSL #2 ; r1 = 5\*sum  ADD r1,r3,r1,LSL #1 ; sum = sum\*10 + r;  LDR r2,=sum  **STR r1,[r2] ; save sum**  MOV r2,#10  SDIV r0,r4,r2 ; r0 = n/10  BL isPal ; recursive call  done **LDR r1,=sum**  **LDR r1,[r1]**  CMP r1,r4  BNE no  yes MOV r0,#1 ; if palindrome  B exit  no MOV r0,#0 ; if not palindrome  exit POP {r4,pc}  ENDP    AREA myData, DATA  ALIGN  sum DCD 0  END |