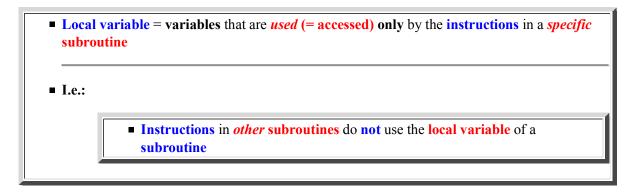
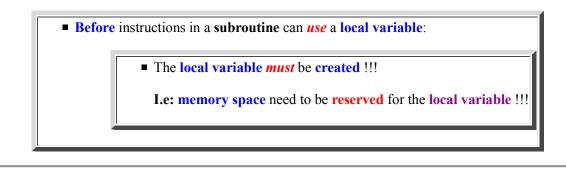
Intro to subroutine with local variables

- What are local variables ?
 - Local variables:



• Fact:



• Review of some CS170 material

• This **should** have been **taught** in **CS170/CS171**, but I want to make **sure** that you **know** *exactly* what happens when a function is **invoked**:

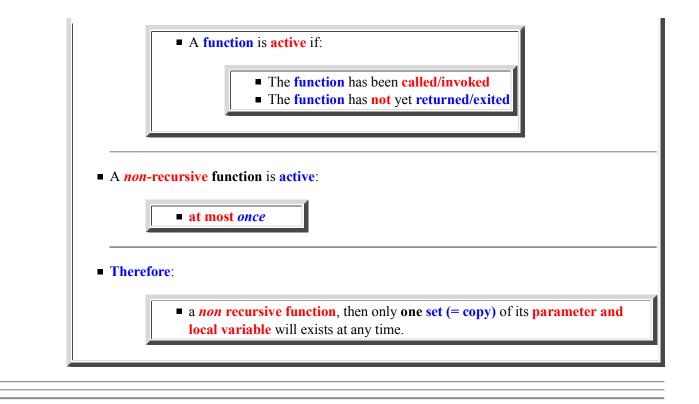
Each time a function/method is invoked (called):

the *parameter* variables and the *local* variables of the (called) function are *created*These variables (parameter and local) are then *destroyed* when:

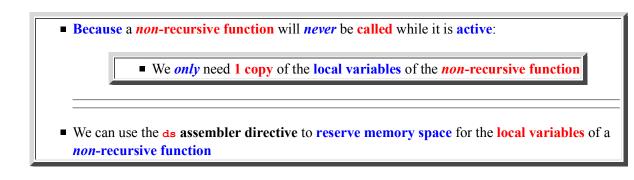
the function exits/returns

• Furthermore:

Active function:



- Local variable of non-recursive function
 - Fact:



• Example: sum all elements in an array

```
int SumArray(int a[], int n)
{
    int i, s; // <-- local variables
    sum = 0;
    for (i = 0; i < n; i++)
        s = s + a[i];
    return(s);
}</pre>
```

This function is **called** by **main()** as follows:

```
main()
{
```

	<pre>int A[10], sum;</pre>	
}	<pre>sum = SumArray(A, 10</pre>);

• I will keep thing **simple** and **pass** the **parameters** using **registers**:

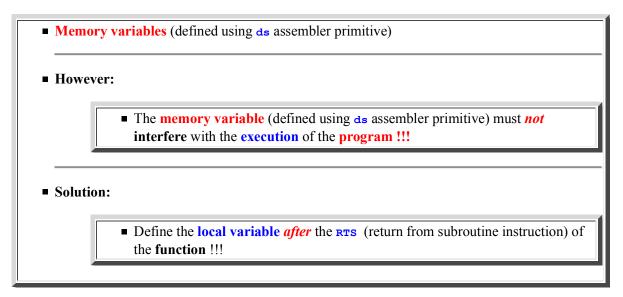
- First parameter is an array. You can't pass multiple integers. The only choice is to pass the address of the array. Let's pick D0. (It's a smarter choice to pick A0 because an address is pass).
- Second parameter can be a constant. So you must pass by value. Let's pick D1.
- And don't forget the **return value** location: let's pick D0.
- Now we write the code in assembler with these agreements on parameters and return location.

First, this is the **main program** that invokes **SumArray**:

main:	
move.1 #A, d0	<pre>// Pass address of array</pre>
move.1 #10, d1	1 // Pass #elements
bsr SumArra	ay // Invoke SumArray
move.l d0, sum	n // When SumArray return, update // total with return value
A: ds.l 10 sum: ds.l 1	// The array

• Then we must decide where to put the local variables

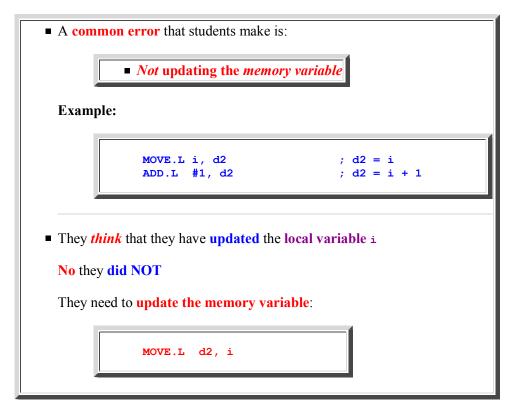
Recall: for a non-recursive function, we can use:



• Solution: Non-recursive function using memory local variables

```
SumArray:
              MOVE.L #0, i ; i
MOVE.L #0, s ; s
     WStart:
              MOVE.L i, d2 ;; Get i in D2
CMP.L d1, d2 ; compares n (d1) and i (d2)
                                           ; if (i >= n) exit while loop
              BGE
                     WEnd
              MOVE.L d0, a0 ; get base addr of array in a0
MOVE.L i, d4 ; d4 = i
              MULS #4, d4
                                           ; d4 = offset in array
              MOVE.L 0(a0, d4.w), d4 ; d4 = a[i]
              MOVE.L s, d3
              ADD.L d4, d3
                                           ;
              MOVE.L d3, s
                                          ; s = s + a[i]
              MOVE.L i, d2 ; d2 = i
ADD.L #1, d2 ; d2 = i + 1
MOVE.L d2 i : i = i + 1
              MOVE.L d2, i
                                           ; i = i + 1
              BRA WStart
       WEnd:
              MOVE.L d3, d0
                                          ; return(s) [ in agreed place d0 ]
              RTS
***** Function will not execute pass this point ****
                         ; reserve SPACE for local variable i
  i:
             ds.l 1
              ds.1 1
  s:
                                      ; reserve SPACE for local variable s
```

```
NOTE:
```



• Here is a runnable Emacsim assembler program of the program above: click here

• Problems with storing local variables using the as directive

• Fact:

• There is only *one* copy of the local variables defined using ds

• We will see **later** (soon) that:

Recursion requires (need to use) one copy of local variables for each invocation of the recursive subroutine

• Therefore:

Local variables stored as memory variables using ds can not support recursive subroutines

We need a more *advance* way to store the local variables for a subroutine !!!

Before I can discuss this technqiue, I want to review the *lifetime* of local variables (and parameter variables)

(I want to make sure you **understand** that **local variables** and **parameter variables** are *created* and *destroyed* while a **program** is running....)

• Historical note

• Fact:

- The *very* first computer language was Fortran
- Fortran did not support recursion !!!!
- The very first Fortran compiler allocate local variables as memory variables --- just like the example above !!!!