How to Call Assembly Subroutines from C...

C calls to assembly subroutines use registers and the stack (if necessary) to pass parameters. In most cases, the first four 8/16-bit parameters of a C function are passed to the assembly code in registers r12, r13, r14, and r15 respectively. Assembly subroutines return a value in r12. Larger data types and/or C functions with more than four parameters use the stack. Variable argument lists (such as with printf) also pass values on the stack as well as in registers.

For example, the C function **int f(w,x,y,z)** would use the following argument to register mapping:

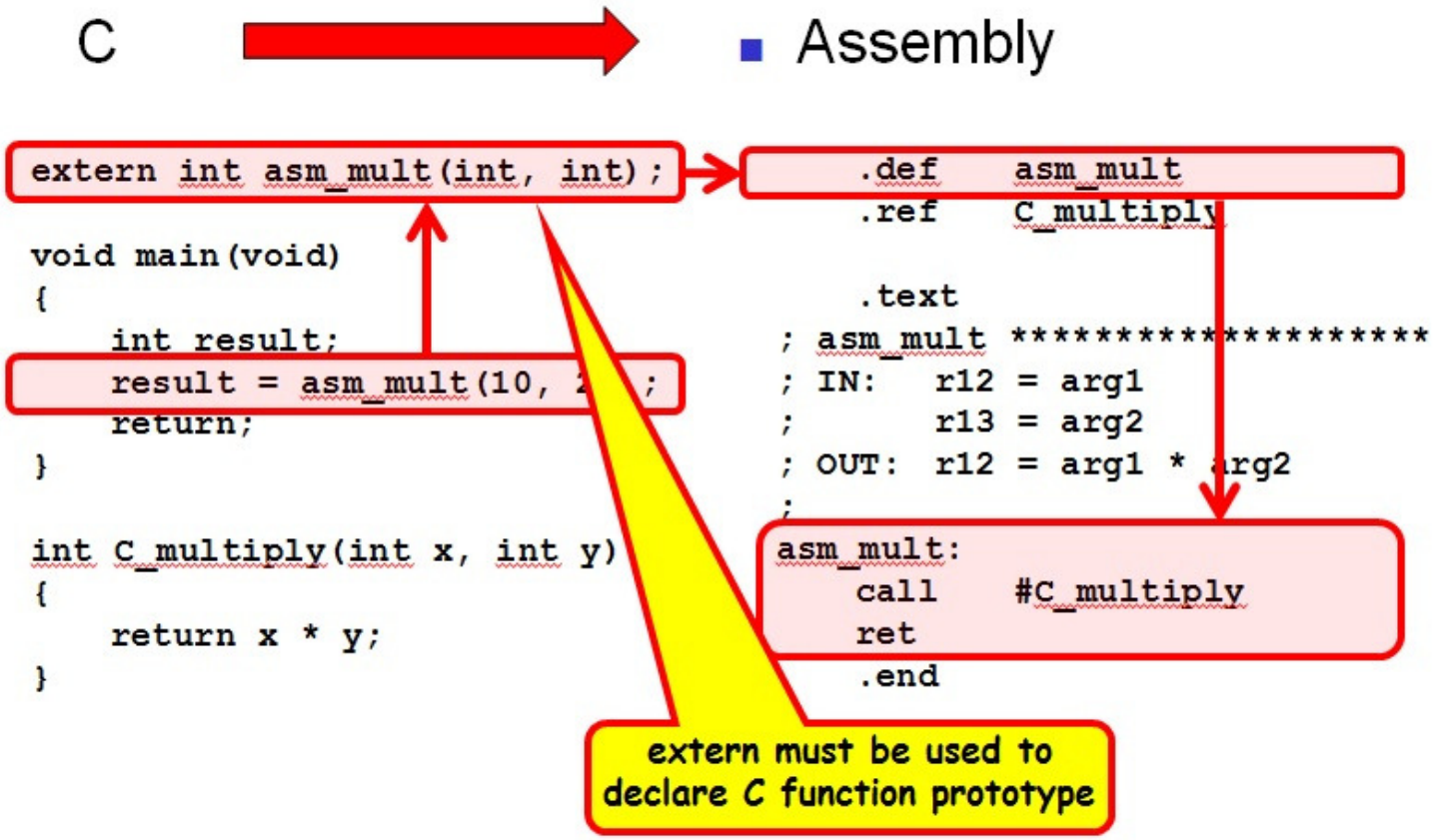
Argument	< 32-bit Type	32-bit Type	struct/union
4th (z)	R15	On the stack	On the stack
3rd (y)	R14	On the stack	On the stack
2nd (x)	R13	R15:R14	On the stack
1st (w)	R12	R13:R12	On the stack
Return Value	R12	R13:R12	Special Area

- Arguments are evaluated right to left.
- If 8 or 16-bit data types, **w**, **x**, **y**, and **z** are passed to the function in registers r12, r13, r14, r15 respectively.
- If 32-bit data types, **z** is pushed onto the stack, followed by , followed by **w** and **x** in registers pairs r13:r12 and r15:r14 respectively.
- The result is returned in R12 (or R13:R12 for a 32 bit type) or in a special area pointed to by R12 if it is a struct or union type.

Remember to make assembly subroutines callee-save:

- The scratch registers R12 to R15 are used for parameter passing and hence are not normally preserved across the call.
- The other general-purpose registers, R4 to R11, are mainly used for register variables and temporary results and must be preserved across a call (callee-save). (This is handled automatically within C.)

Finally:



Function parameters The parameters of a called function are passed to an assembler routine in a right to left order. Up to four left most parameters are passed in registers unless they are defined as a struct or union type, in which case they are also passed on the stack. The remaining parameters are always passed on the stack. <-->