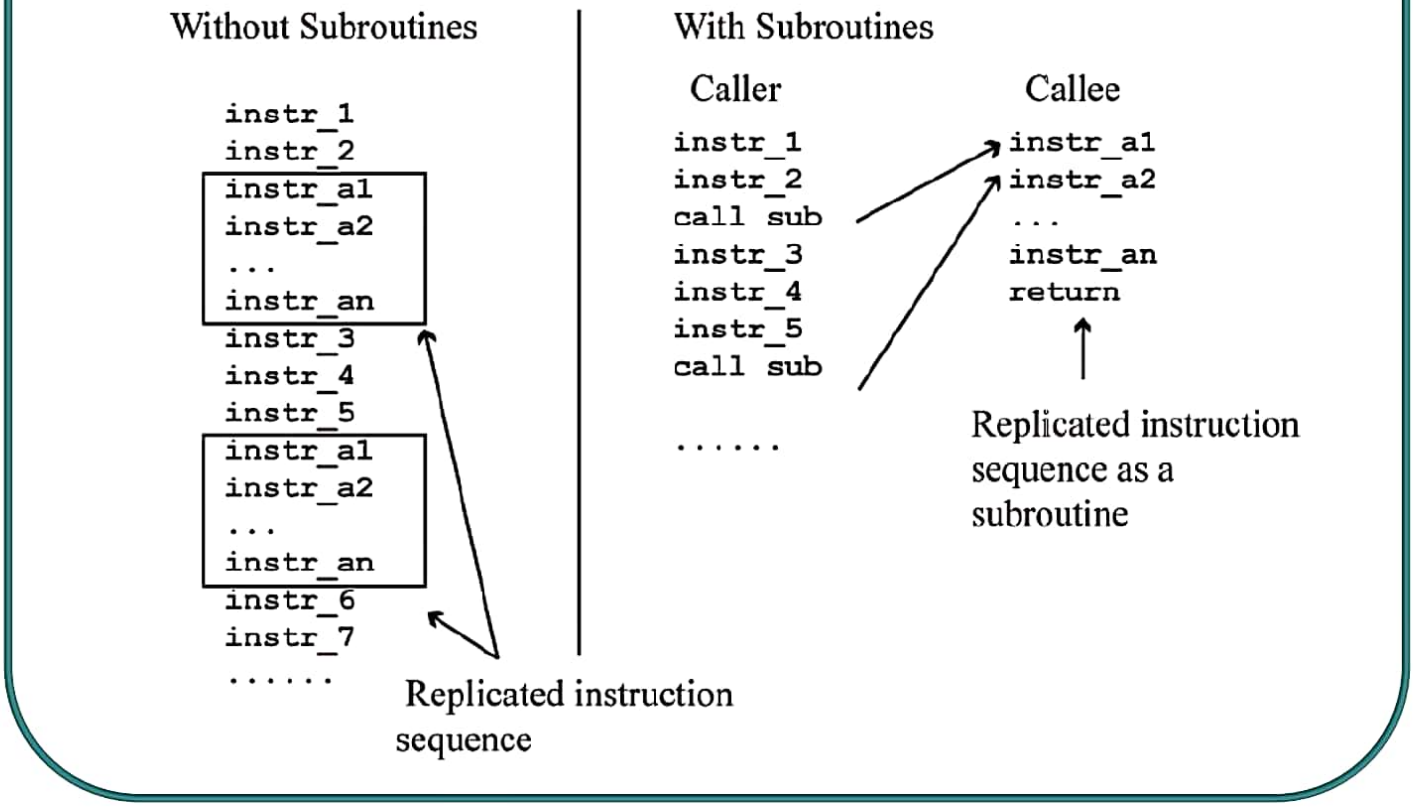
Basic Idea

- Large programs are hard to handle
  - We can break them to smaller programs
  - They are called subroutines
- Subroutines are called from the main program
- Writing subroutines
  - When should we jump? (use CALL)
  - Where do we return to? (use RETURN)

Subroutine

- A **subroutine** is a block of code that is **called** from different places from within a main program or other subroutines.
- Saves code space in that the subroutine code does not have to be repeated in the program areas that need it;
  - Only the code for the subroutine call is repeated.
- A subroutine can have
  - **parameters** that control its operation
  - **local variables** for computation.
- A subroutine may pass a **return value** back to the caller.
- Space in data memory must be reserved for parameters, local variables, and the return value.

Subroutine

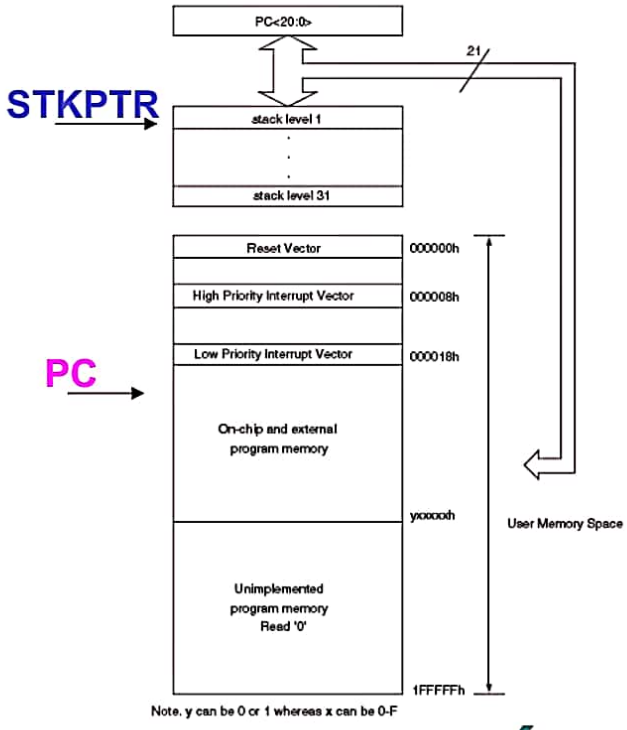


Using Subroutines

- When using subroutines we need to know the following:
  - Where is the NEXT instruction's address
  - How to remember the RETURN address
- Subroutines are based on MPU instructions and use STACK

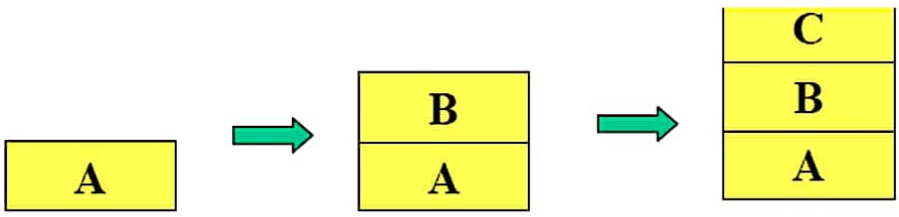
Stack

- Temporary memory storage space used during the execution of a program
- Used by MPU
- Stack Pointer (SP)
  - The MPU uses a register called the stack pointer, similar to the program counter (PC), to keep track of available stack locations.



Data Storage via the Stack

- The word 'stack' is used because storage/retrieval of words in the stack memory area is the same as accessing items from a stack of items.
- Visualize a stack of boxes. To build a stack, you place box A, then box B, then box C
  - Notice that you only have access to the last item placed on the stack (the Top of Stack –TOS). You retrieve the boxes from the stack in reverse order (C then B then A). A stack is also called a LIFO (last-in-first-out) buffer (similar to a Queue)



Instructions to Store and Retrieve Information from the Stack

- PUSH
  - Increment the memory address in the stack pointer (by one) and stores the contents of the counter (PC+2) on the top of the stack
- POP
  - Discards the address of the top of the stack and decrement the stack pointer by one
- The contents of the stack (21-bit address), pointed by the stack pointer, are copied into three special function registers
  - TOSU (Top-of-Stack Upper), TOSH (High), and TOSL (Low)

Subroutine Call

- In the PIC18F, the stack is used to store the **return address** of a subroutine call.
- The return address is the place in the calling program that is returned to when subroutine exits.
- On the PIC18Fxx, the return address is PC+4, if PC is the location of the call instruction .
  - Call is a 2-word instruction!
- The return address is PC+2 if it is a rcall instruction.

CALL Instruction

- CALL Label, S (0/1) ;Call subroutine
  ; located at Label
- CALL Label, FAST ;FAST is equivalent to
  ; S = 1
  - If S = 0: Increment the stack pointer and store the contents of the program counter (PC+4) on the top of the stack (TOS) and branch to the subroutine address located at Label.
  - If S = 1: Increment the stack pointer and store the contents of the program counter (PC+4) on the top of the stack (TOS) and the contents of W, STATUS, and BSR registers in their respective **shadow registers**, and branch to the subroutine address located at Label.

RETURN Instruction

- RETURN,0 → gets the address from TOS and moves it to PC, decrements stack pointer
- RETURN,1 → gets the address from TOS and moves it to PC, decrements stack pointer; retrieves all shadow registers (WREG, STATUS, BSR)*
- RETLW → gets the address from TOS and moves it to PC ; returns literal to WREG, decrements stack pointer

* 1 or FAST

Subroutine Architecture

How do we write a subroutine?