Chapter 04: Instruction Sets and the Processor organizations

Lesson 13

Subroutine Nesting Using Stacks to Implement Subroutine Calls

Objective

- To understand call and return
- Nested calls

Subroutine

Subroutine

- Also called a routine
- A set of instructions or sub-program provided for a specific purpose

Examples of Subroutine programs

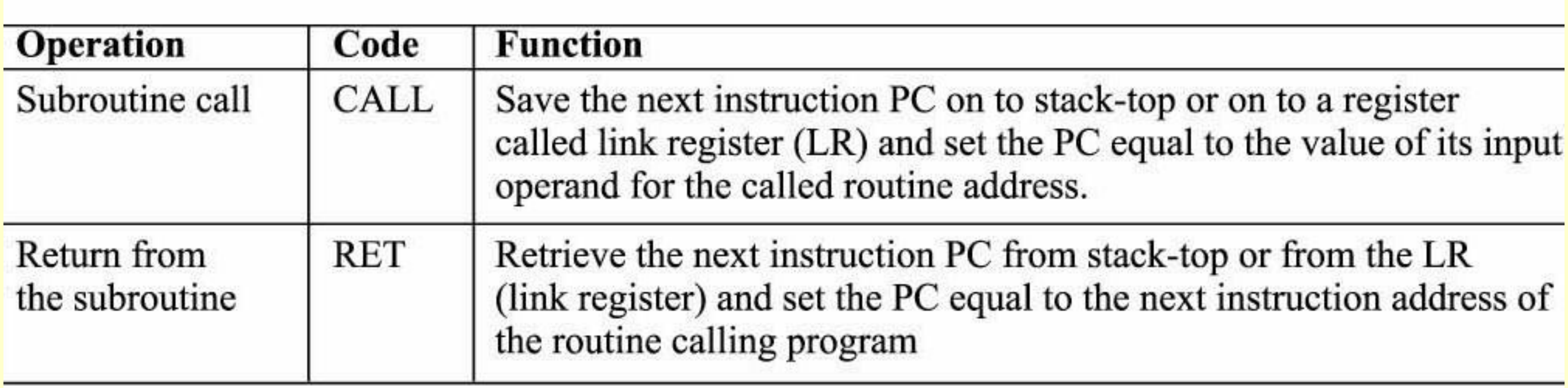
- Delay according to some parameter and the parameter passed as input to the routine
- Cube of a parameter and the parameter passed as input to the routine
- Sum of N- numbers in a table with the values of N and table start-address passed as inputs to the routine

Return instruction

- A program calls the subroutine by an instruction CALL
- At the last instruction in the subroutine, there is RET instruction for return to the calling program

Subroutine related instructions in instruction set

Subroutine CALL and RET instructions



Operation	Code	Function
Subroutine call	CALL	Save the next instruction PC on to stack-top or on to a register called link register (LR) and set the PC equal to the value of its input operand for the called routine address.
Return from the subroutine	RET	Retrieve the next instruction PC from stack-top or from the LR (link register) and set the PC equal to the next instruction address of the routine calling program

Sequence of program instructions at StepK of a program

- StepK: (instruction *k*);
- (instruction *k* + 1);
- (instruction *k* + 2);
- CALL SBR_X;
- .
- StepL: SBR_X: (instruction *l*)
- . .
- ; PC for StepL saves at stack-top or LR

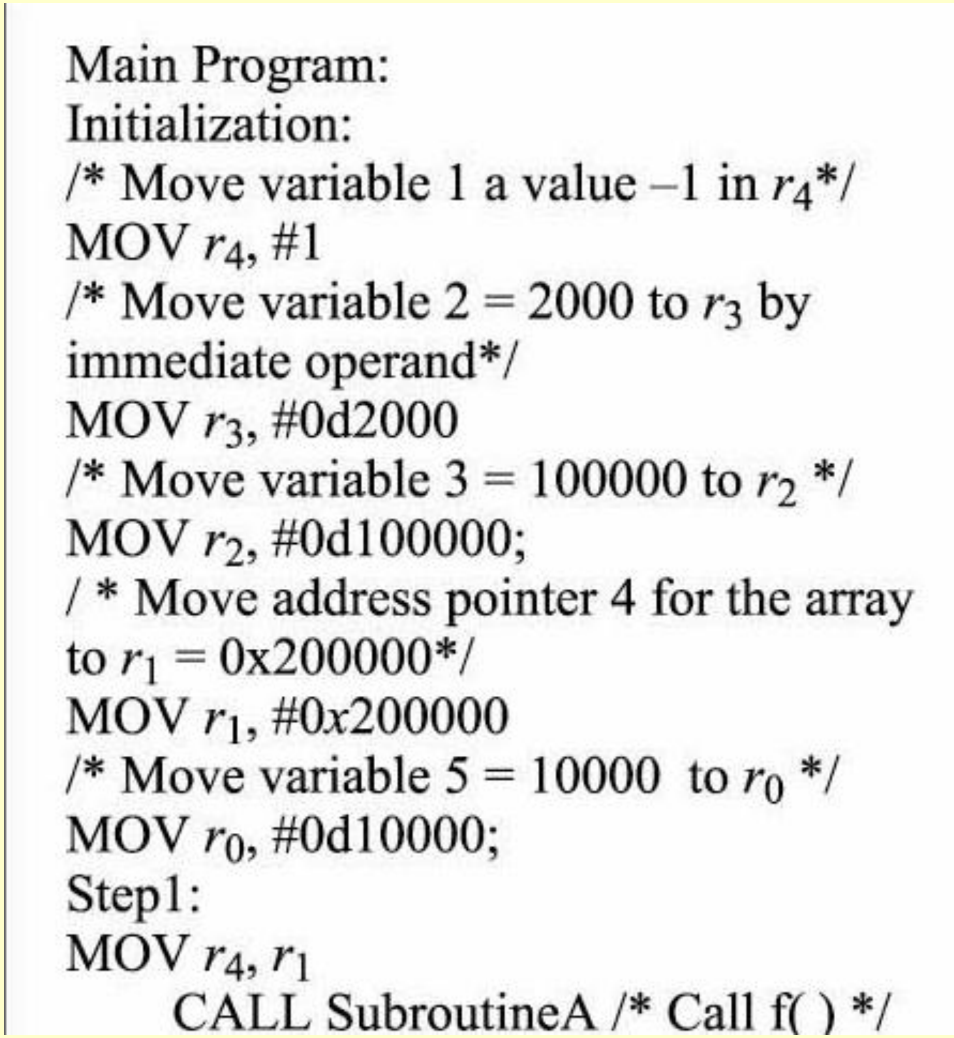
and PC gets the address of SBR_X after the CALL instruction

Sequence of program instructions at SBR_X routine (a sub-program)

- SBR_X: (instruction *l*)
- .
- .
- RET
- ; PC gets the address for StepL from the stack-top or from the LR after the execution of RET instruction

Example of Subroutines and their call from a main program

Main program sequence of instructions and call for subroutine f ( ) in Step 1



Main Program:

Initialization:

/* Move variable 1 a value –1 in r_4*/

MOV r_4, #1

/* Move variable 2 = 2000 to r_3 by immediate operand*/

MOV r_3, #0d2000

/* Move variable 3 = 100000 to r_2 */

MOV r_2, #0d100000;

/ * Move address pointer 4 for the array to r_1 = 0x200000*/

MOV r_1, #0*x*200000

/* Move variable 5 = 10000 to r_0 */

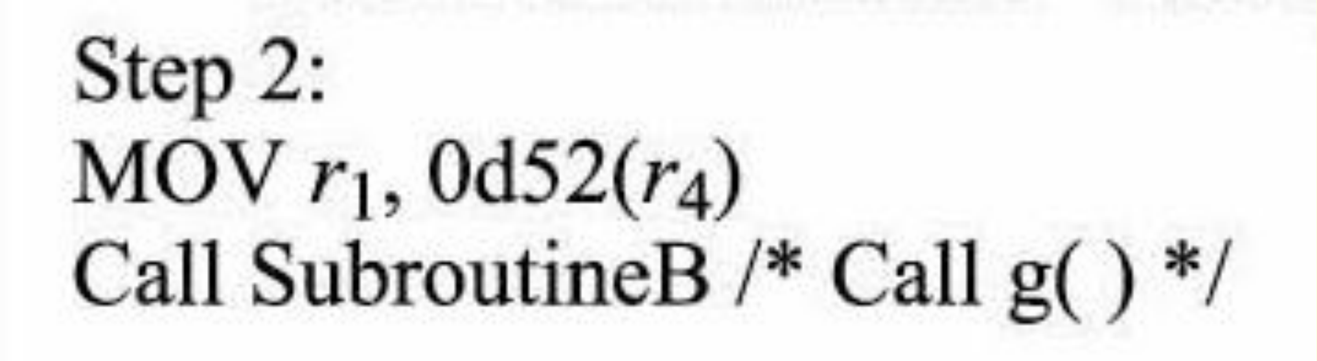
MOV r_0, #0d10000;

Step1:

MOV r_4, r_1

    CALL SubroutineA /* Call f( ) */

Main program sequence of instructions and call for subroutine g ( ) in Step 2

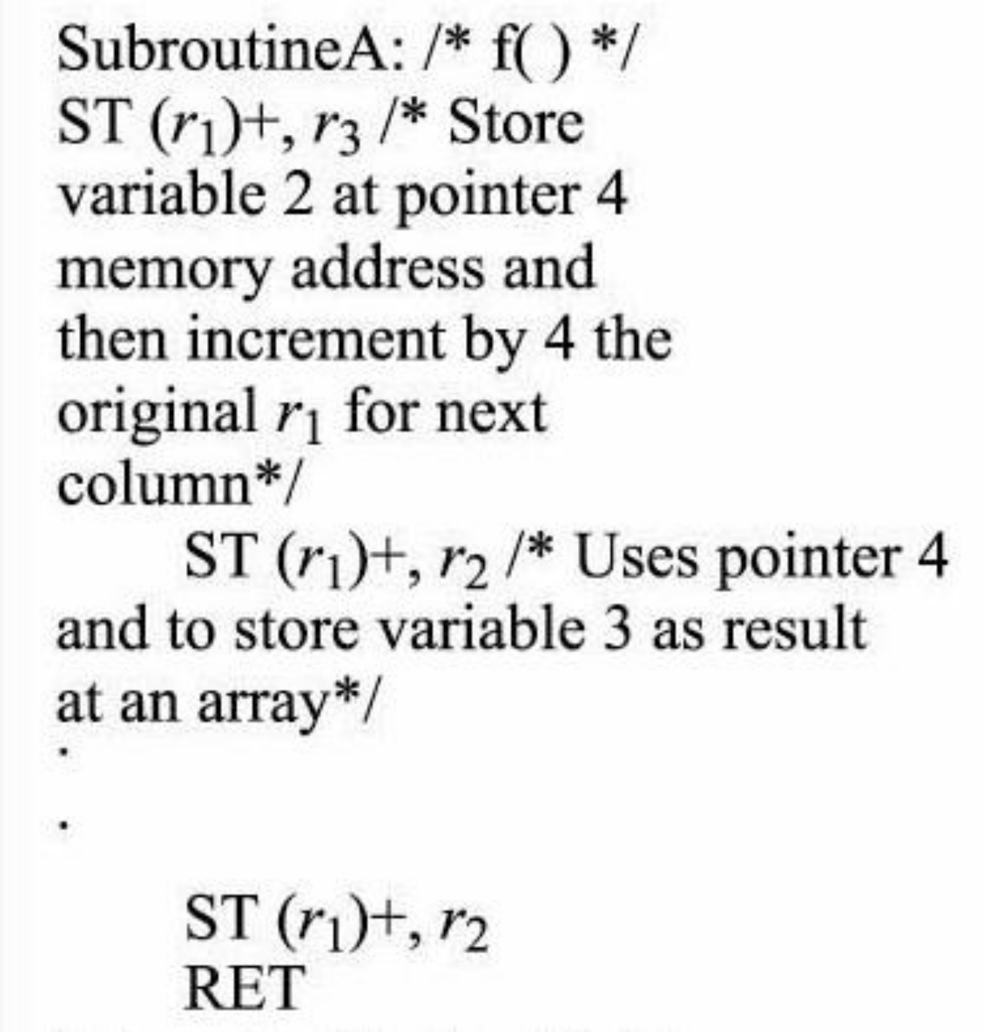


Step 2:

MOV r_1, 0d52(r_4)

Call SubroutineB /* Call g( ) */

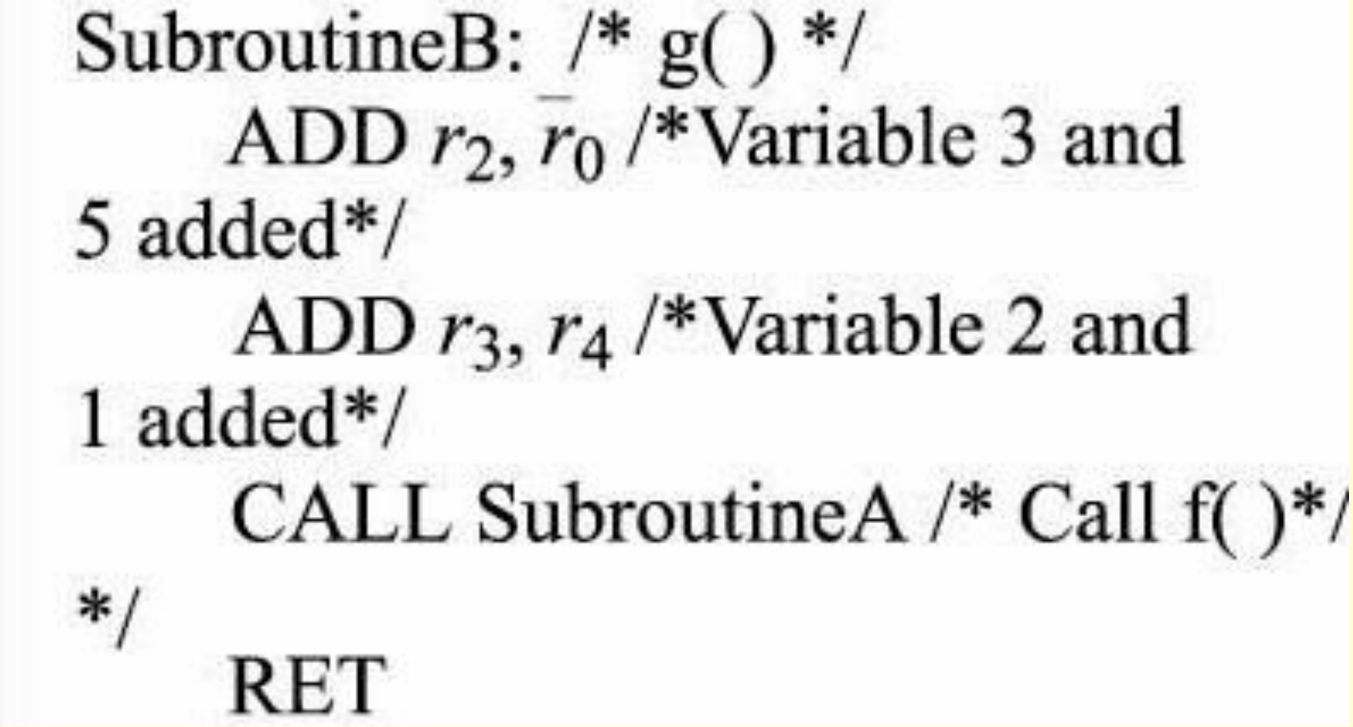
Sequence of instructions in subroutine f ( ) and RET at the end



SubroutineA: /* f( ) */
ST (r_1)+, r_3 /* Store variable 2 at pointer 4 memory address and then increment by 4 the original r_1 for next column*/
        ST (r_1)+, r_2 /* Uses pointer 4 and to store variable 3 as result at an array*/
.
.

        ST (r_1)+, r_2
        RET

Sequence of instructions in subroutine g ( ) and RET at the end



SubroutineB: /* g( ) */
        ADD r_2, r_0 /*Variable 3 and
5 added*/
        ADD r_3, r_4 /*Variable 2 and
1 added*/
        CALL SubroutineA /* Call f( )*/
*/
        RET

Passing of the Parameters and Reference to parameters to Subroutine on call

Parameter passing though registers

- Efficient
- But when the number of parameters (input and output) is too large, the number of registers at the processor may not be enough
- Parameter can be passed from one routine to another by passing by values as well as by references

Parameters and reference in the example

- r0, r2, r3 and r4 are variables that are passed to subroutine B by the values
- Main program passes pointer 4 at r1

Passing of the Parameters

- Main program passes to subroutine f() the input variables 2 and 3
- Main program passes to subroutine g() the input variables 1, 2, 3 and 5 through r4, r3, r2 and r0, respectively
- g() passes the results through r2 and r3 when it calls f()

Passing of the reference to Parameters

- r1 is a pointer and it thus passes memory reference address to subroutine A

Nesting of Subroutine calls

Nesting of calls

- A subroutine calling one subroutine and that calling another before the Return
- For example, Main program calls subroutine B and Subroutine B calls subroutine A before return
- Since PC always saves at top of the stack the return is always to the calling routine

Summary

We learnt

- Subroutine call by CALL instruction
- Return by RET instruction
- PC saves on call at stack top or link register
- PC returns back from stack top or link registers
- Parameters are passed by value or references to parameters
- Nesting of calls

End of Lesson 13 on

Subroutine Nesting Using Stacks to Implement Subroutine Calls