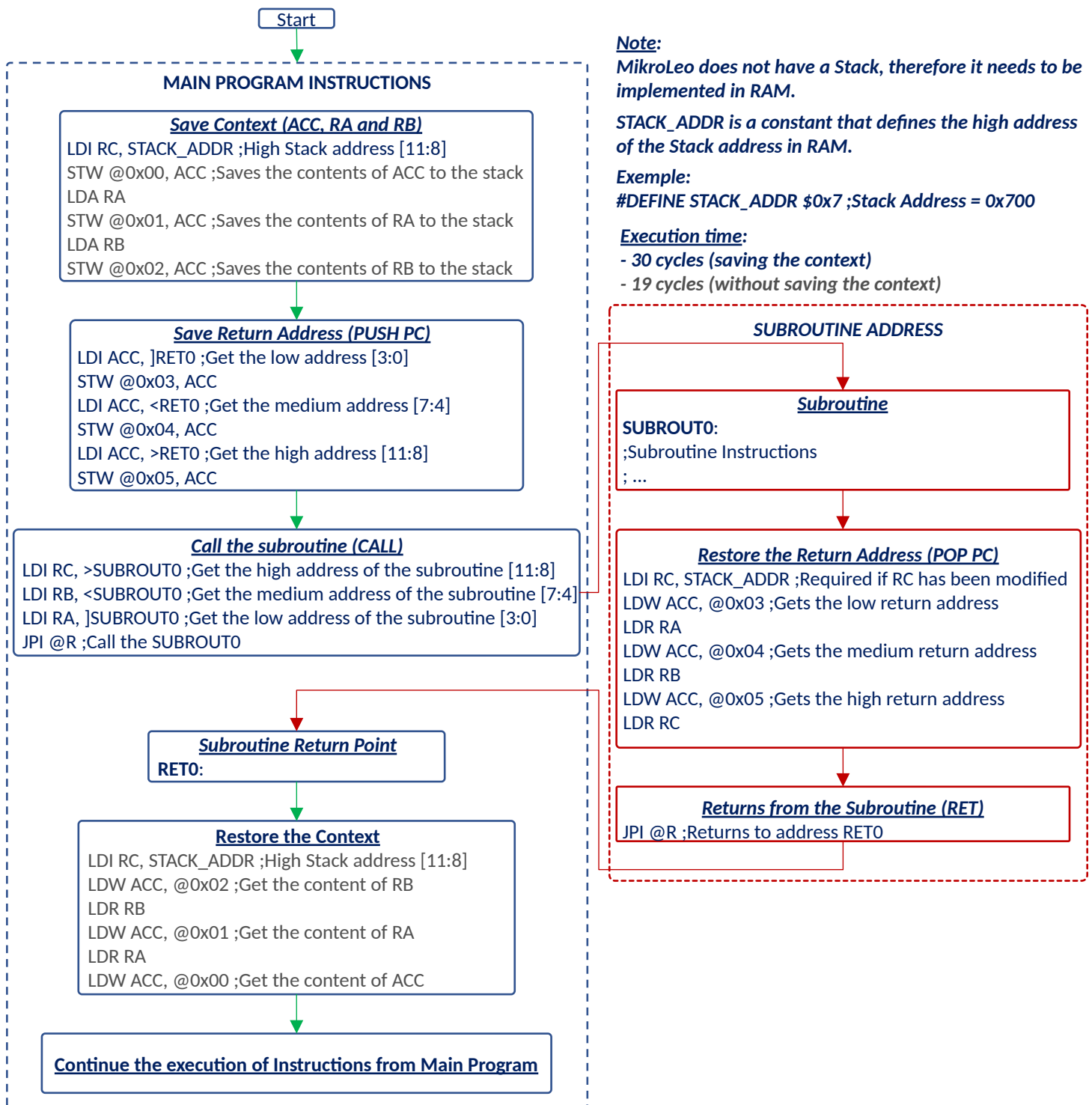


Where is the stack in the MikroLeo?

Flowchart and code example for implementing a simple subroutine (1-level Stack) using indirect addressing



Note:

The method uses six nibbles of RAM to save the current context (the contents of the ACC, RA, and RB) and to store the subroutine's return address (the RET0 address). Of these, three nibbles of RAM are used to store the program memory address corresponding to the RET0 label. In this example, the initial stack address is STACK_ADDR:0:0, which corresponds to address 700h. Thus, the contents of the stack in RAM have the following structure:

0x700 = ACC	Address 700h in RAM stores the contents of the ACC
0x701 = RA	Address 701h in RAM stores the contents of the RA
0x702 = RB	Address 702h in RAM stores the contents of the RB
0x703 = RET0[3:0]	Address 703h in RAM stores the low address of RET0
0x704 = RET0[7:4]	Address 704h in RAM stores the medium address of RET0
0x705 = RET0[11:8]	Address 705h in RAM stores the high address of RET0

Example of how code is stored in program memory and how a subroutine is manipulated using indirect addressing

ROM Address	Instruction Word	Instruction Mnemonic
0x000	3007	INI: LDI RC,STACK_ADDR
0x001	0a00	STW @0x00,ACC
0x002	1300	LDA RA
0x003	0a01	STW @0x01,ACC
0x004	2300	LDA RB
0x005	0a02	STW @0x02,ACC
0x006	0000	LDI ACC,]RET0
0x007	0a03	STW @0x03,ACC
0x008	0001	LDI ACC,<RET0
0x009	0a04	STW @0x04,ACC
0x00A	0000	LDI ACC,>RET0
0x00B	0a05	STW @0x05,ACC
0x00C	3000	LDI RC,>SUBROUT0
0x00D	2001	LDI RB,<SUBROUT0
0x00E	1008	LDI RA,]SUBROUT0
0x00F	4c00	JPI @R
0x010	3007	RET0: LDI RC,STACK_ADDR
0x011	0202	LDW ACC,@0x02
0x012	2700	LDR RB
0x013	0201	LDW ACC,@0x01
0x014	1700	LDR RA
0x015	0200	LDW ACC,@0x00
0x016	3000	LDI RC,>INI
0x017	0c00	JPI INI
0x018	3007	SUBROUT0: LDI RC,STACK_ADDR
0x019	0203	LDW ACC,@0x03
0x01A	1700	LDR RA
0x01B	0204	LDW ACC,@0x04
0x01C	2700	LDR RB
0x01D	0205	LDW ACC,@0x05
0x01E	3700	LDR RC
0x01F	4c00	JPI @R

Save context on the STACK

Save address of RET label on STACK

STACK

Get the address of the subroutine and Call it

Return Address

Restore the context from the STACK

Execute the subroutine code, restore the return address from the STACK (RET0), and return to the main program

RAM Address	Content
000h	...
001h	...
003h	...
⋮	⋮
700h	ACC
701h	RA
702h	RB
703h	0
704h	1
705h	0
706h	
707h	
708h	
709h	
70Ah	
70Bh	
70Ch	
⋮	⋮
7FDh	...
7FEh	...
7FFh	...

RET0[3:0]

RET0[7:4]

RET0[11:8]

Note: Note that in this example, for simplicity, there is no subroutine code. The subroutine code must begin at the address SUBROUT0.