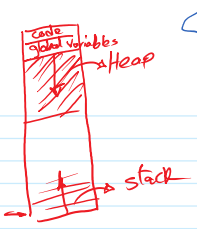
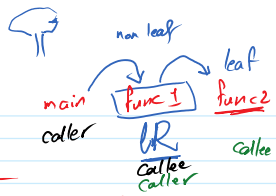
Tutorial 5

Tuesday, April 27, 2021 11:00 AM





BL label
LR next instruction address

Q1. When does the LR have to be pushed on the stack?

in non leaf functions. i.e functions that call other functions. to preserve its value since calling another function would overwrite the LR value we need to return to.

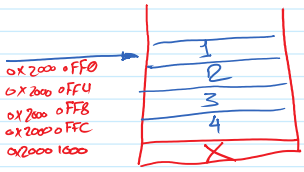
leaf fn
non leaf fn

$a0, ...$a3
$v0, $v1

parameter
result

R4 : R11 Preserve ones

Q2. Show the SP value and the content of stack after executing this instruction PUSH{R4, R6-R8} assuming the SP initially equals 0x2000.1000 and R4=1, R6=2, R7=3, R8=4. What will be the values of the registers R0-R4 after executing this instruction POP{R0-R3}?



R0=1, R1=2, R2=3, R3=4

Push {R0}

STR R0, [SP, #-4]!

low address

SP=SP-4

SP

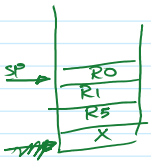
16 bit

Push { }

high address

R4, R3, R2
R2 - R4

Push {R5, R1, R0}
Push {R1, R5, R0}
Push {R0, R1, R5}



Q3. Explain how does the return from subroutine work in these two functions?

Function PUSH {R4,LR}	Function2
;stuff	;stuff
POP {R4,PC}	BX LR

LR BL Function

Push R4, LR to the stack

branch update Program counter using LR value

Pop R4, PC

return address from stack → put Program counter return function.

Q4. Write assembly code that pushes registers R1, R3, and R5 onto the stack. (in order)

Push R1
Push R3
Push R5

Push {R1, R3, R5}

Q5. What are the addressing modes used in each of the following instructions?

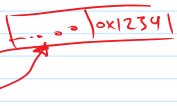
(Register Indirect) Indexed Addressing

LDR R0, [R1] Indexed addressing
LDR R2, [R1, #4] " " with immediate offset
MOV R3, #100 Immediate addressing
BL function PC relative
MOV R0, #1 Immediate addressing
LDRB R0, [PC, #0x30] PC relative
LDR R0, =1234567 pseudo instruction ⇒ PC-relative addressing

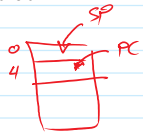
PC relative
LDR R0, =0x12345
LDR R0, [PC, #8]
...
DCD #0x12345
Immediate addressing

~~MOV #0x1234~~

MOV R0, #0x1234
MOVT R0, #0xABCD



Q6. Write a complete ARM assembly program that calls the procedure func1 which in turn calls a procedure func2. The procedure func2 is defined in Q1 of Sheet 3.



Reset_Handler
...
BL func1
...
halt
B halt
func1
Push {LR}

CO:

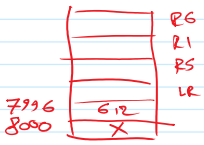
16. Consider the following C-language program. Functions main(), calculate(), evaluate(), and square() are placed starting at locations 600_{10}, 700_{10}, 800_{10}, and 900_{10}, respectively in memory. Integer variables c, d, and i are stored in registers ~~$s0, $s1, and $s2~~ R5, R6, and R7, respectively. Assume that register ~~$s0, $s1~~ R5, R6, and the stack pointer (~~$sp~~ SP) are initialized with the values 6, 9, and 8000_{10}, respectively. Show stack contents (in decimal) when the stack is at its maximum size while executing the given code. Do not write register names. Write only numerical contents. Use X for unknown values.

```
main () {
        int i= calculate(2,5);
        ...
        ...
}

int calculate (int x, int y) {
        return(evaluate (x+y,8));
}

int evaluate (int a, int b) {
        int c= square(a) + b;
        return c;
}

int square (int u) {
        int d= u * u;
        return d;
}
```



Embedded Page 2