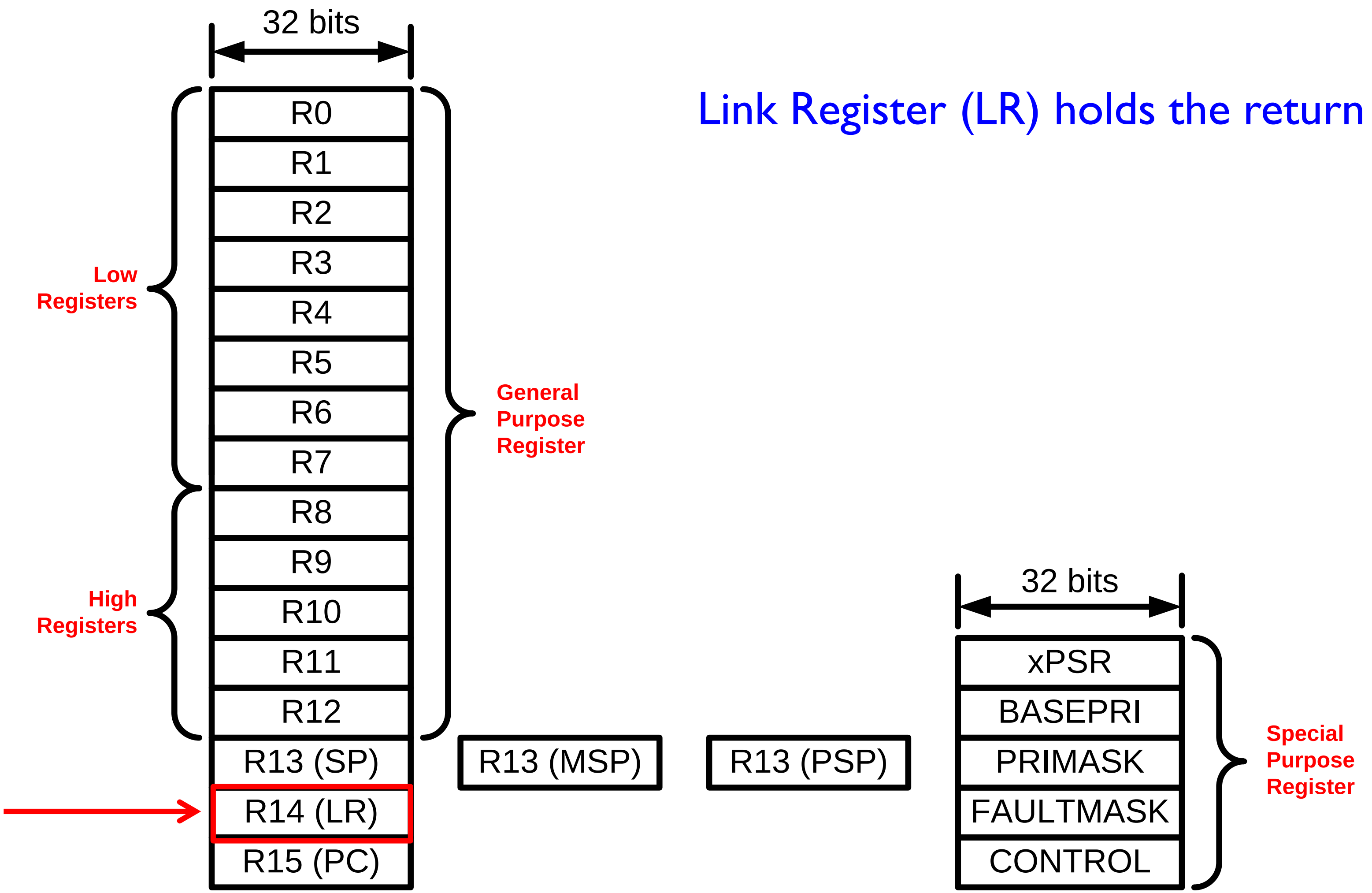
Embedded Systems with ARM Cortex-M3 Microcontrollers in Assembly Language and C

Chapter 8
Passing Parameters to Subroutines via Registers

Dr. Yifeng Zhu
Electrical and Computer Engineering
University of Maine

Spring 2015

Link Register



Link Register (LR) holds the return address

Calling a Subroutine

BL *label*

- `Step 1: LR = PC + 4`
- `Step 2: PC = label`

- Notes:
  - *label* is name of subroutine
  - Compiler translates label to memory address
  - After call, LR holds return address (the instruction following the call)

Caller Program

```
    MOV r4, #100
    ...
    BL  foo
    ...
```

Subroutine/Callee

```
foo PROC
    ...
    MOV   r4, #10
    ...
    BX    LR
    ENDP
```

Exiting a Subroutine

BX *LR*

- PC = LR

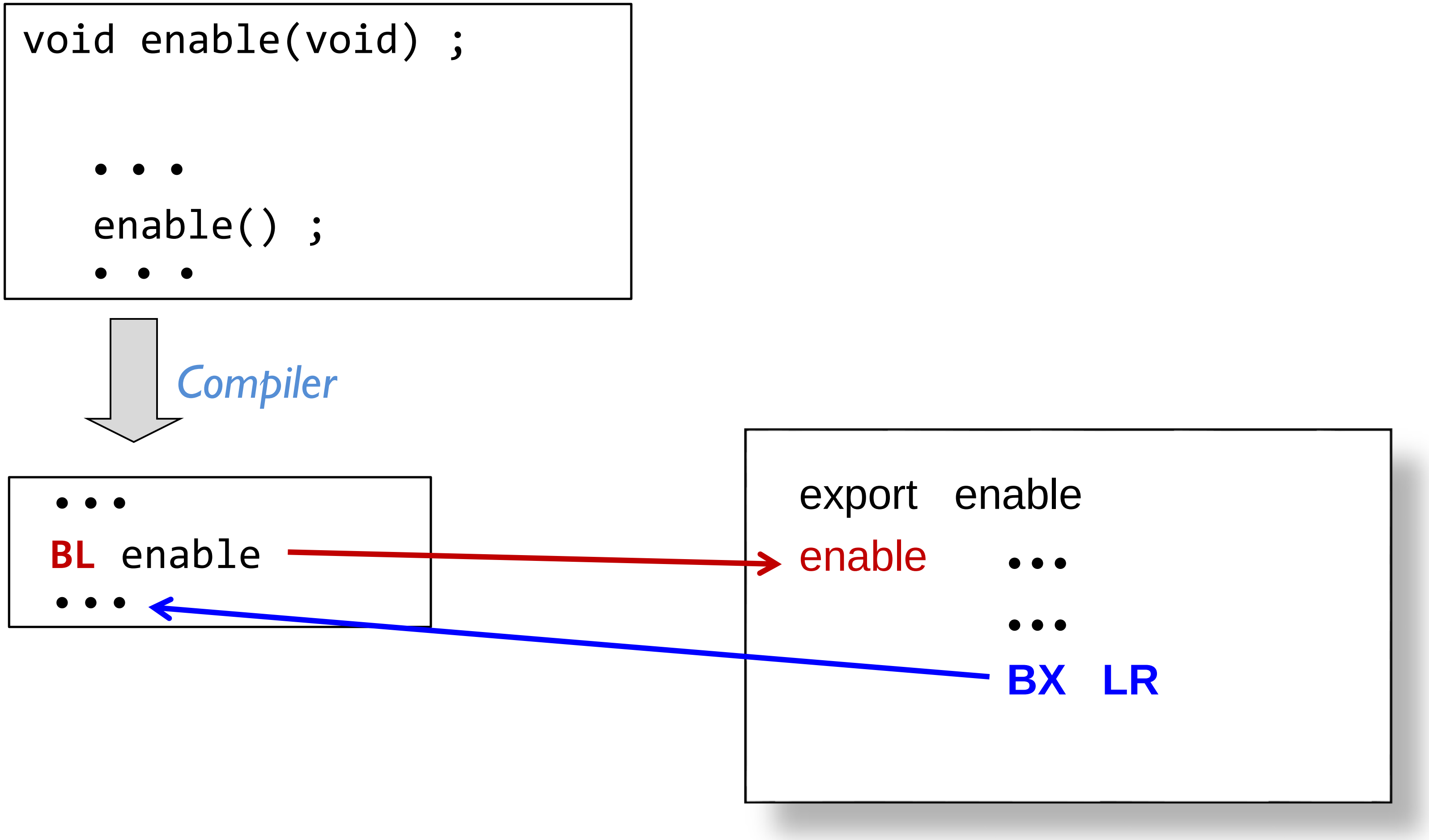
Caller Program

```
    MOV r4, #100
    ...
    BL  foo
    ...
```

Subroutine/Callee

```
foo PROC
    ...
    MOV    r4, #10
    ...
    BX     LR
    ENDP
```

BL and BX



```
void enable(void) ;


    • • •
    enable() ;
    • • •
```

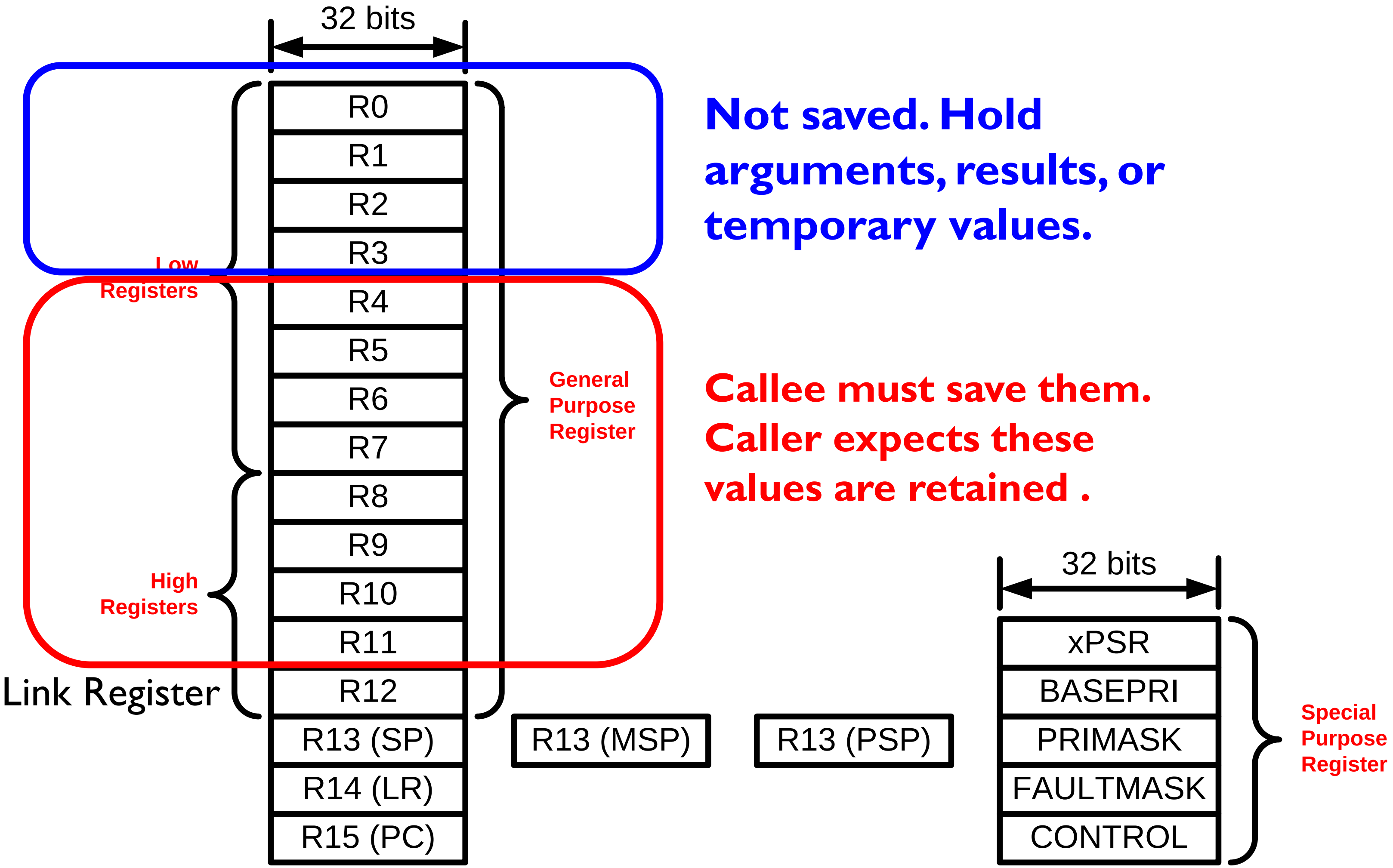
```
  • • •
  BL  enable
  • • •
```

```
export   enable
enable      • • •
            • • •
            BX   LR
```

ARM Procedure Call Standard

Register	Usage	Subroutine Preserved	Notes
r0	Argument 1 and return value	No	If return has 64 bits, then r0:r1 hold it. If argument 1 has 64 bits, r0:r1 hold it.
r1	Argument 2	No	
r2	Argument 3	No	If the return has 128 bits, r0-r3 hold it.
r3	Argument 4	No	If more than 4 arguments, use the stack
r4	General-purpose V1	Yes	Variable register 1 holds a local variable.
r5	General-purpose V2	Yes	Variable register 2 holds a local variable.
r6	General-purpose V3	Yes	Variable register 3 holds a local variable.
r7	General-purpose V4	Yes	Variable register 4 holds a local variable.
r8	General-purpose V5	YES	Variable register 5 holds a local variable.
r9	Platform specific/V6	No	Usage is platform-dependent.
r10	General-purpose V7	Yes	Variable register 7 holds a local variable.
r11	General-purpose V8	Yes	Variable register 8 holds a local variable.
r12 (IP)	Intra-procedure-call register	No	It holds intermediate values between a procedure and the sub-procedure it calls.
r13 (SP)	Stack pointer	Yes	SP has to be the same after a subroutine has completed.
r14 (LR)	Link register	No	LR does not have to contain the same value after a subroutine has completed.
r15 (PC)	Program counter	N/A	Do not directly change PC

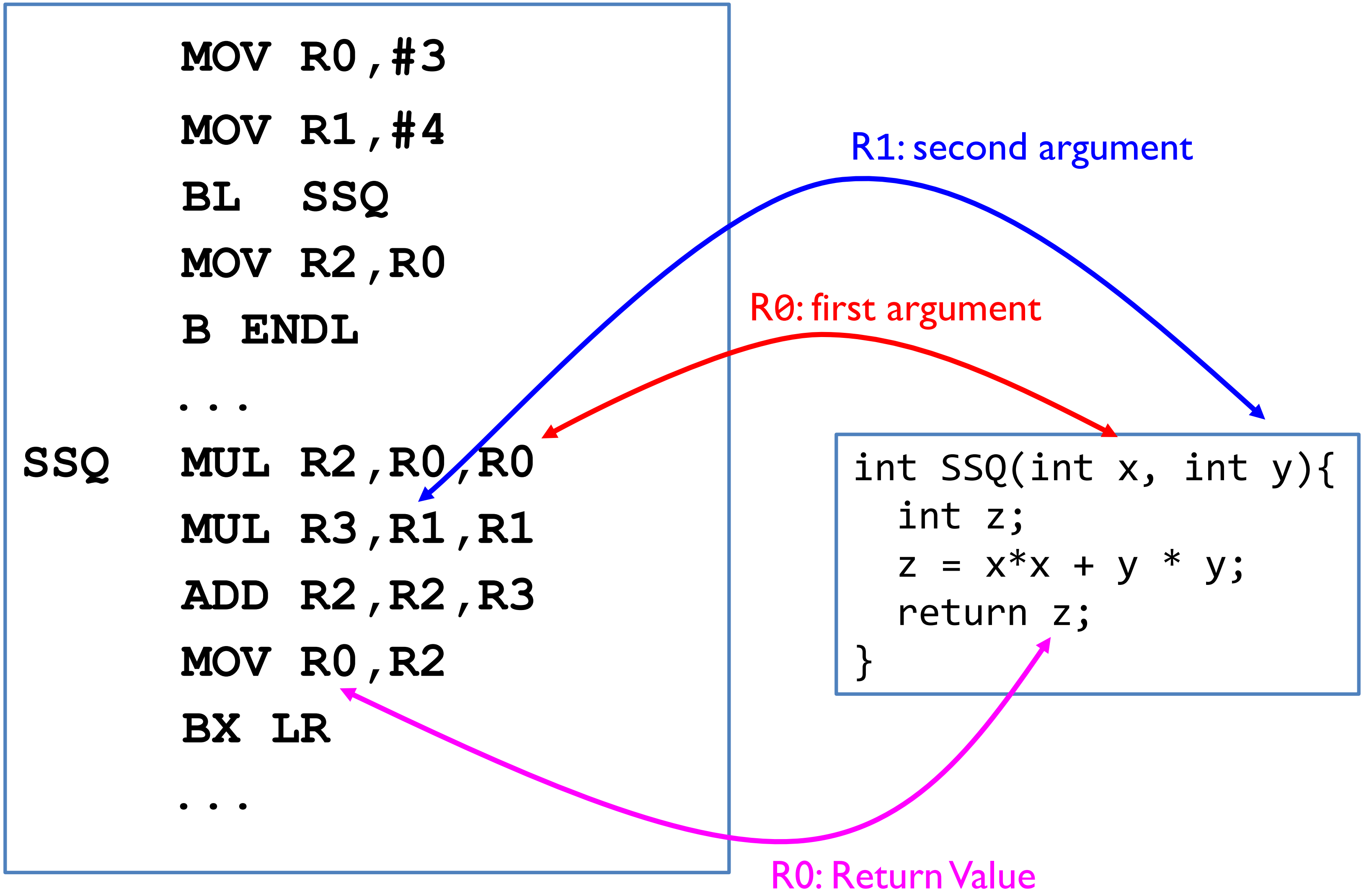
Link Register



Not saved. Hold arguments, results, or temporary values.

Callee must save them. Caller expects these values are retained .

Example: **R2 = R0*R0+R1*R1**



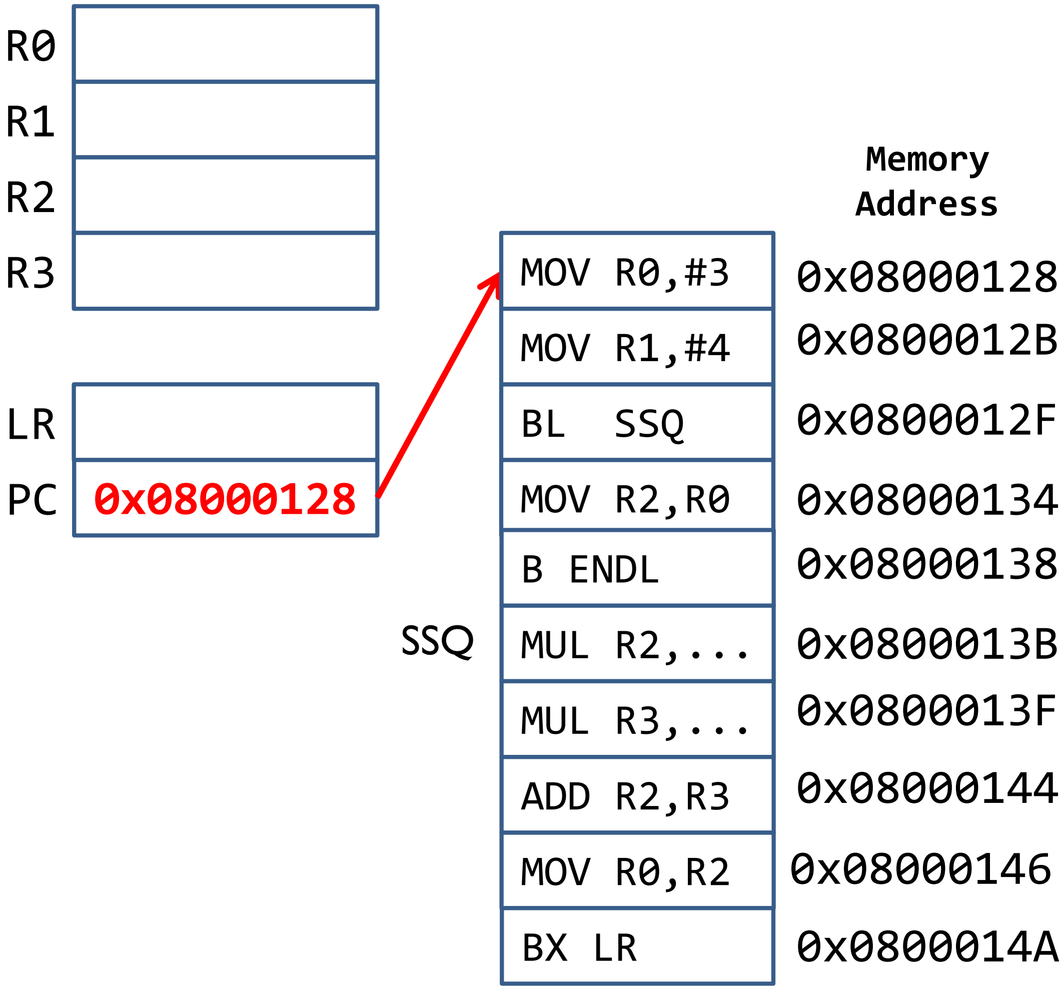
```
      MOV R0,#3
      MOV R1,#4
      BL  SSQ
      MOV R2,R0
      B ENDL
      ...
SSQ   MUL R2,R0,R0
      MUL R3,R1,R1
      ADD R2,R2,R3
      MOV R0,R2
      BX LR
      ...
```

```
int SSQ(int x, int y){
   int z;
   z = x*x + y * y;
   return z;
}
```

Example: R2 = R0*R0+R1*R1

```
        MOV R0,#3
        MOV R1,#4
        BL  SSQ
        MOV R2,R0
        B  ENDL

SSQ     MUL R2,R0,R0
        MUL R3,R1,R1
        ADD R2,R2,R3
        MOV R0,R2
        BX  LR
ENDL    ...
```



R0

R1

R2

R3

LR

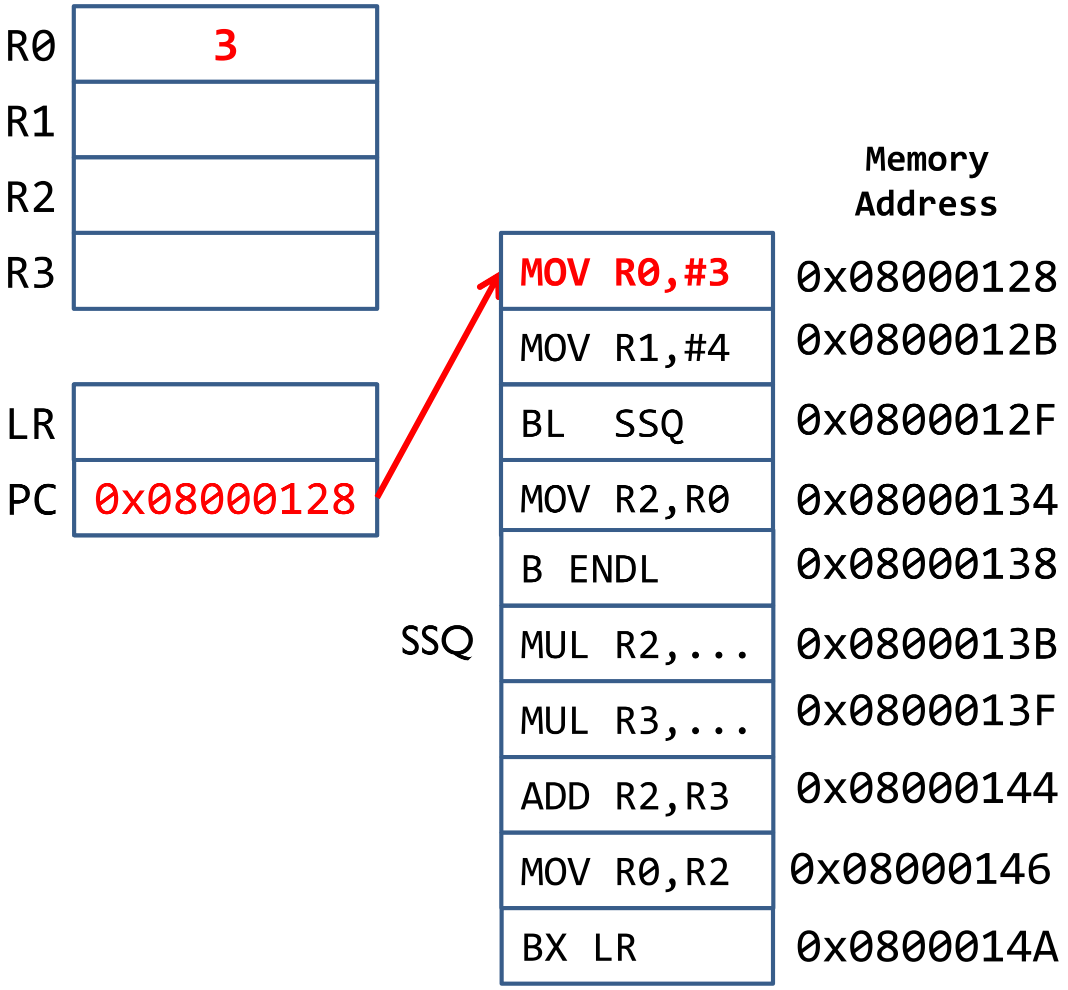
PC 0x08000128

		Memory Address
	MOV R0,#3	0x08000128
	MOV R1,#4	0x0800012B
	BL SSQ	0x0800012F
	MOV R2,R0	0x08000134
	B ENDL	0x08000138
SSQ	MUL R2,...	0x0800013B
	MUL R3,...	0x0800013F
	ADD R2,R3	0x08000144
	MOV R0,R2	0x08000146
	BX LR	0x0800014A

Example: R2 = R0*R0+R1*R1

```
        MOV R0,#3
        MOV R1,#4
        BL  SSQ
        MOV R2,R0
        B ENDL

SSQ     MUL R2,R0,R0
        MUL R3,R1,R1
        ADD R2,R2,R3
        MOV R0,R2
        BX  LR
ENDL    ...
```



R0	3
R1	
R2	
R3	

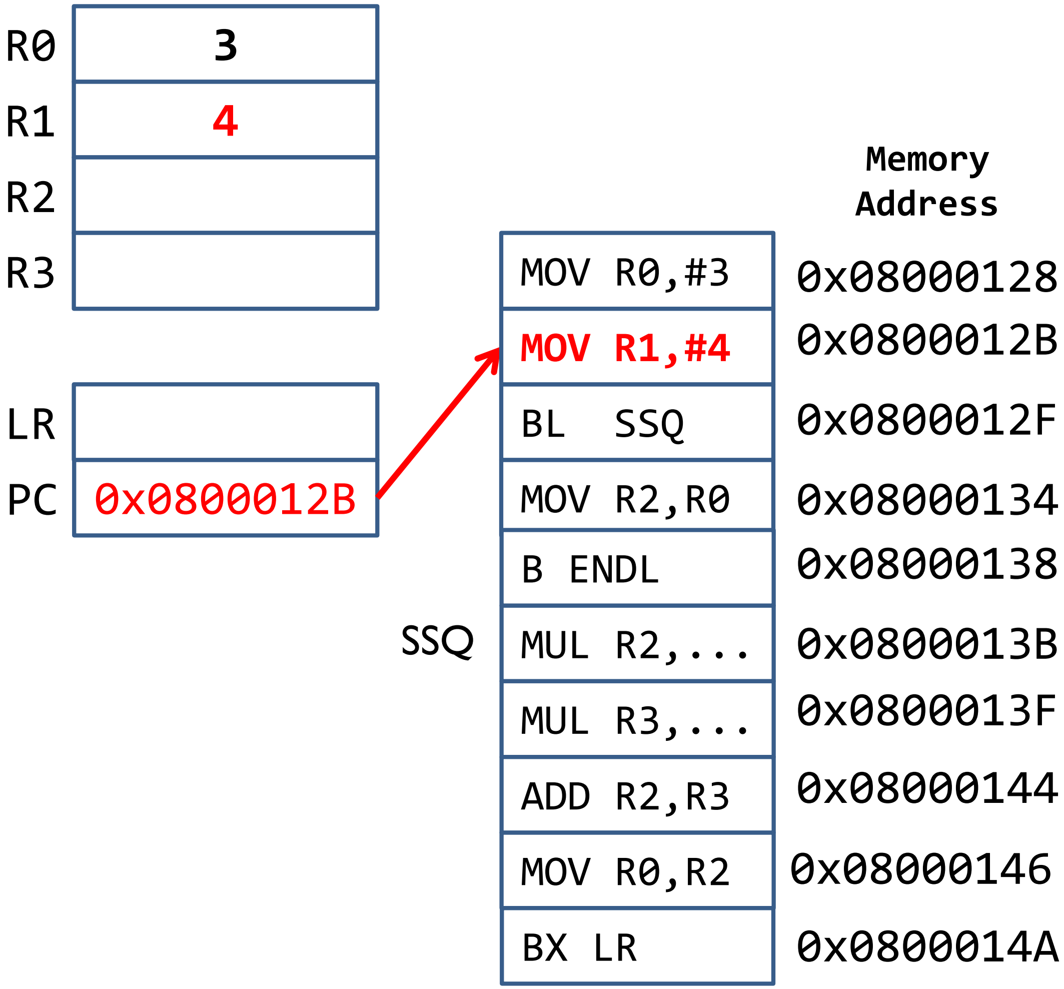
LR	
PC	0x08000128

		Memory Address
	MOV R0,#3	0x08000128
	MOV R1,#4	0x0800012B
	BL SSQ	0x0800012F
	MOV R2,R0	0x08000134
	B ENDL	0x08000138
SSQ	MUL R2,...	0x0800013B
	MUL R3,...	0x0800013F
	ADD R2,R3	0x08000144
	MOV R0,R2	0x08000146
	BX LR	0x0800014A

Example: R2 = R0*R0+R1*R1

```
        MOV R0,#3
        MOV R1,#4
        BL  SSQ
        MOV R2,R0
        B ENDL

SSQ     MUL R2,R0,R0
        MUL R3,R1,R1
        ADD R2,R2,R3
        MOV R0,R2
        BX  LR
ENDL    ...
```



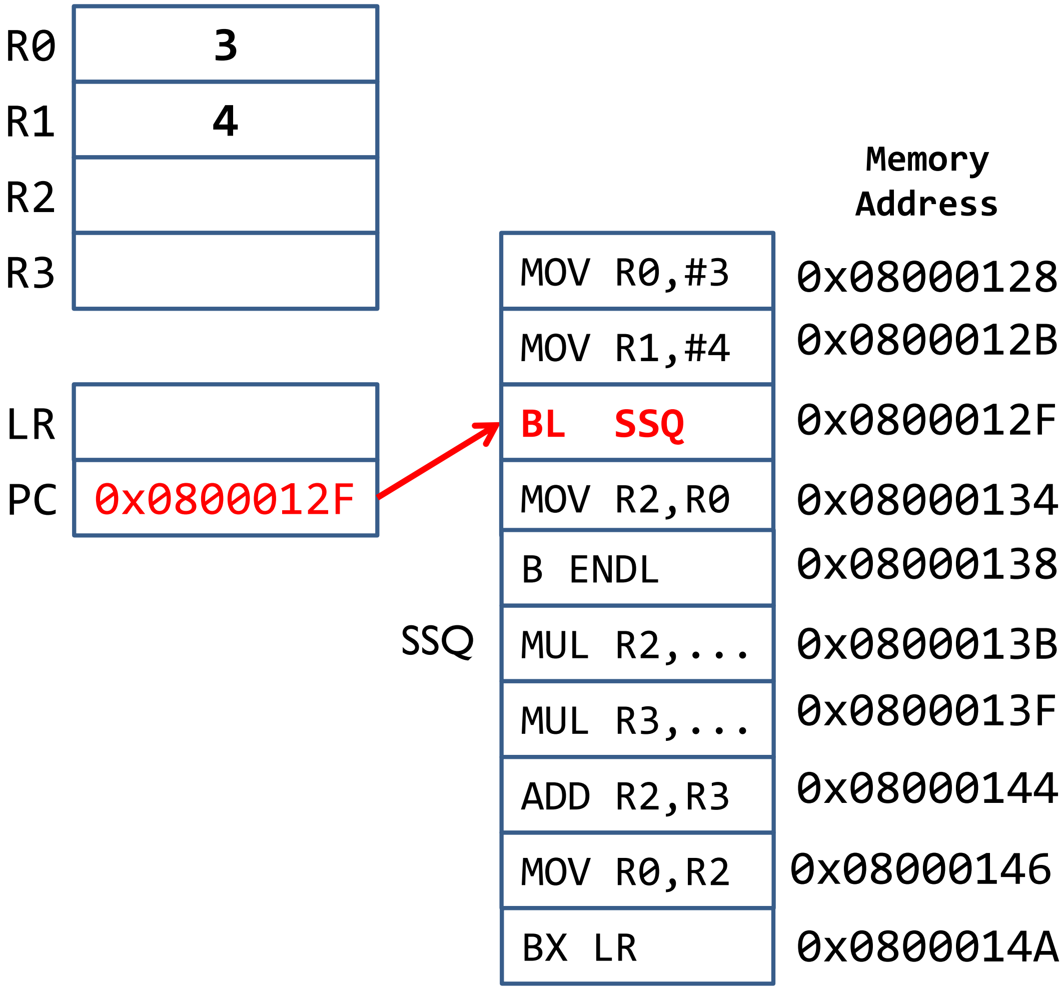
R0	3
R1	4
R2	
R3	
LR	
PC	0x0800012B

		Memory Address
	MOV R0,#3	0x08000128
	MOV R1,#4	0x0800012B
	BL SSQ	0x0800012F
	MOV R2,R0	0x08000134
	B ENDL	0x08000138
SSQ	MUL R2,...	0x0800013B
	MUL R3,...	0x0800013F
	ADD R2,R3	0x08000144
	MOV R0,R2	0x08000146
	BX LR	0x0800014A

Example: R2 = R0*R0+R1*R1

```
        MOV R0,#3
        MOV R1,#4
        BL  SSQ
        MOV R2,R0
        B ENDL

SSQ     MUL R2,R0,R0
        MUL R3,R1,R1
        ADD R2,R2,R3
        MOV R0,R2
        BX  LR
ENDL    ...
```



R0	3
R1	4
R2	
R3	

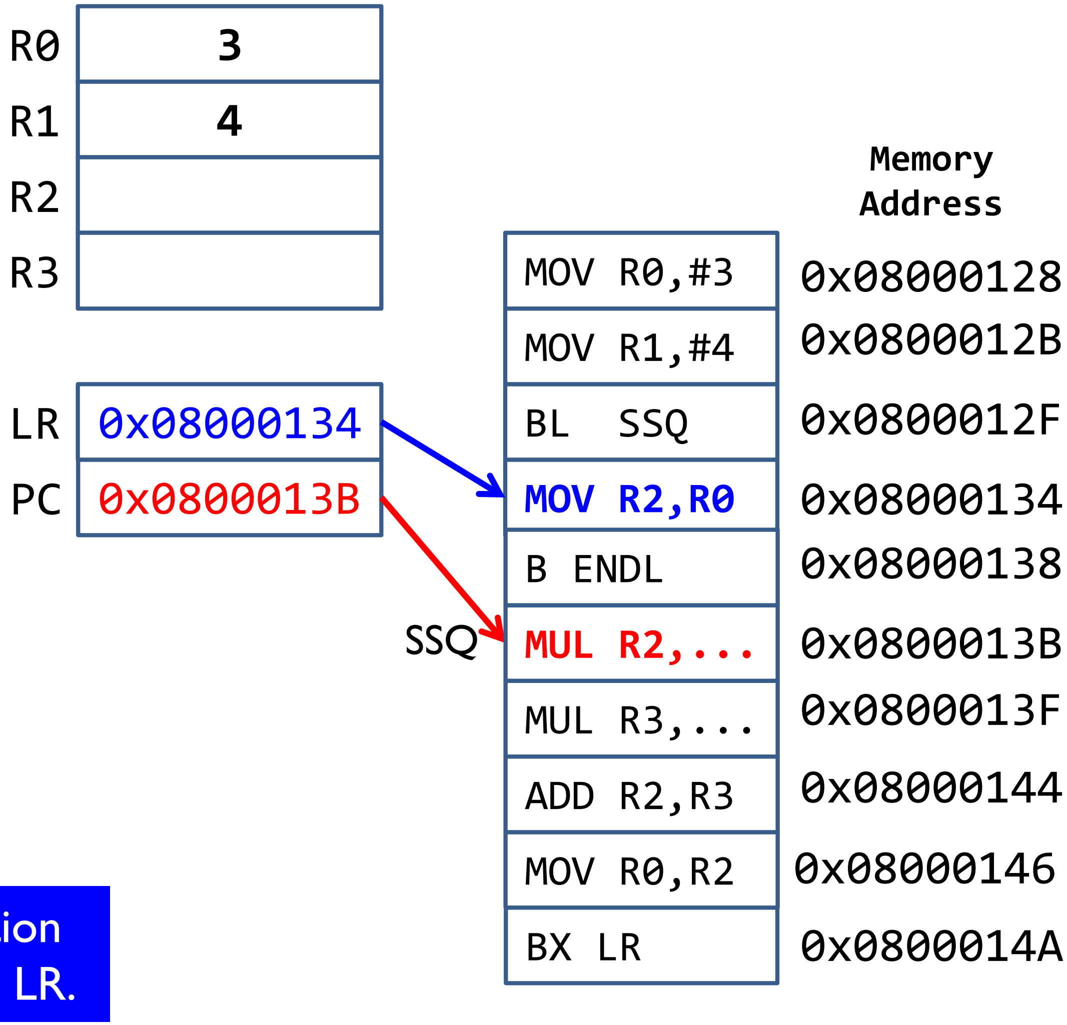
LR	
PC	0x0800012F

		Memory Address
	MOV R0,#3	0x08000128
	MOV R1,#4	0x0800012B
	BL SSQ	0x0800012F
	MOV R2,R0	0x08000134
	B ENDL	0x08000138
SSQ	MUL R2,...	0x0800013B
	MUL R3,...	0x0800013F
	ADD R2,R3	0x08000144
	MOV R0,R2	0x08000146
	BX LR	0x0800014A

Example: R2 = R0*R0+R1*R1

```
        MOV R0,#3
        MOV R1,#4
        BL  SSQ
        MOV R2,R0
        B ENDL

SSQ     MUL R2,R0,R0
        MUL R3,R1,R1
        ADD R2,R2,R3
        MOV R0,R2
        BX  LR
ENDL    ...
```



R0	3
R1	4
R2	
R3	

LR	0x08000134
PC	0x0800013B

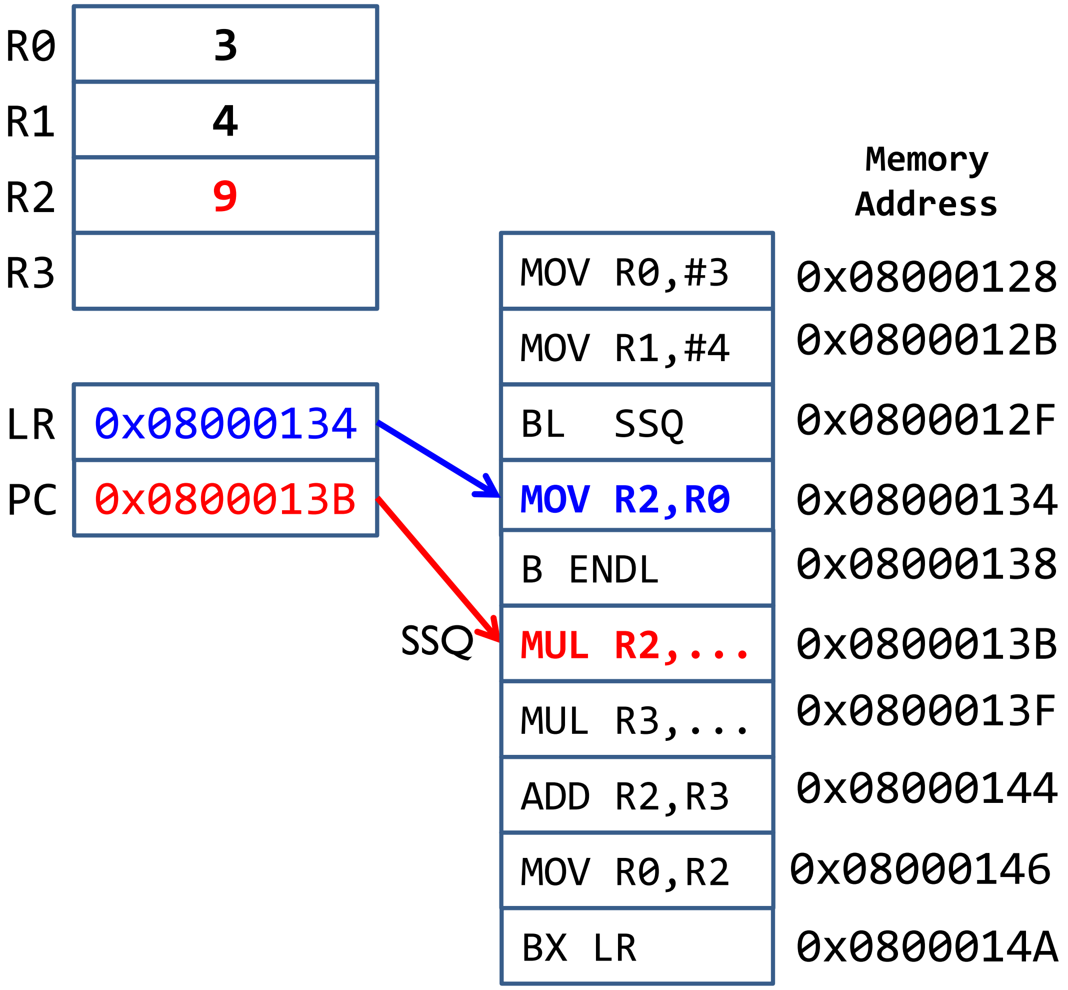
	Memory Address
MOV R0,#3	0x08000128
MOV R1,#4	0x0800012B
BL SSQ	0x0800012F
MOV R2,R0	0x08000134
B ENDL	0x08000138
MUL R2,...	0x0800013B
MUL R3,...	0x0800013F
ADD R2,R3	0x08000144
MOV R0,R2	0x08000146
BX LR	0x0800014A

Address of the next instruction after the branch is saved into LR.

Example: R2 = R0*R0+R1*R1

```
        MOV R0,#3
        MOV R1,#4
        BL  SSQ
        MOV R2,R0
        B  ENDL

SSQ     MUL R2,R0,R0
        MUL R3,R1,R1
        ADD R2,R2,R3
        MOV R0,R2
        BX  LR
ENDL    ...
```



R0	3
R1	4
R2	9
R3	

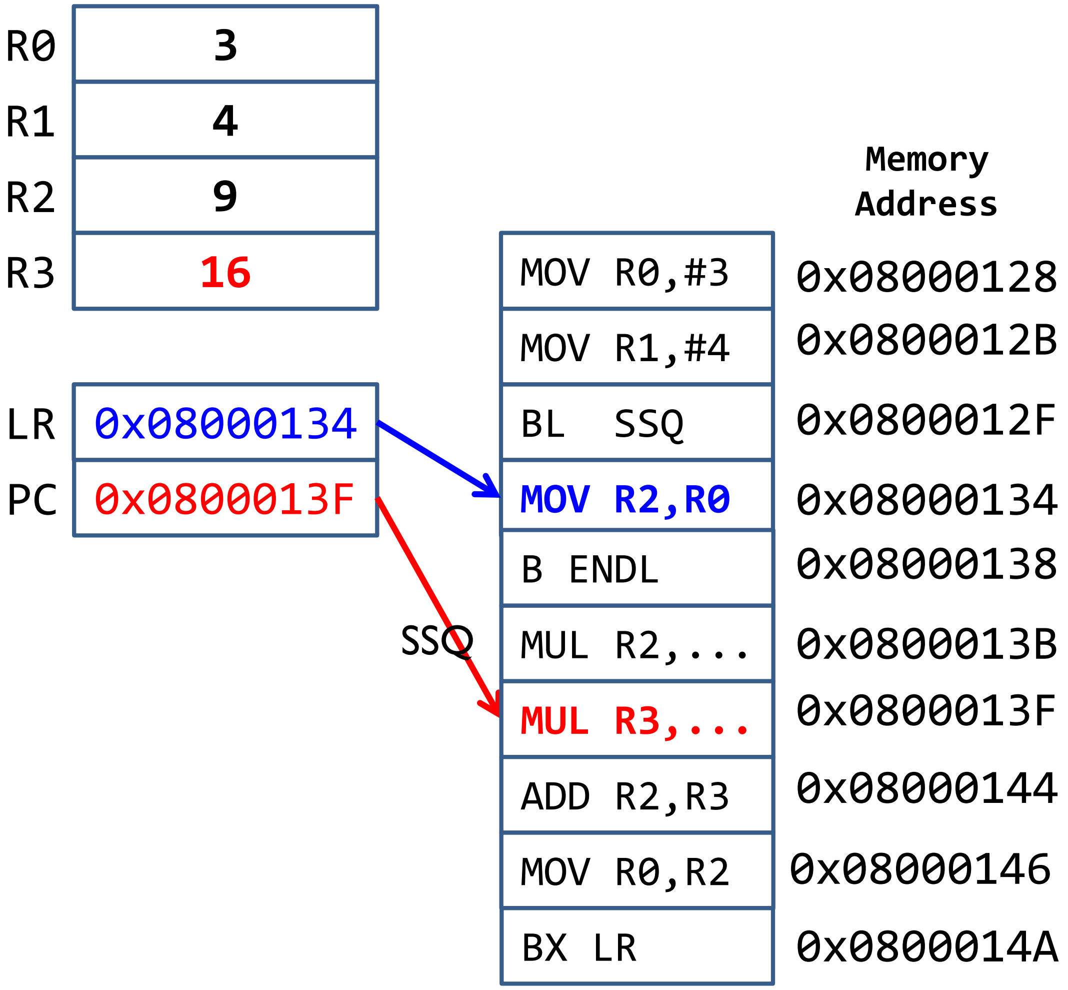
LR	0x08000134
PC	0x0800013B

	Memory Address
MOV R0,#3	0x08000128
MOV R1,#4	0x0800012B
BL SSQ	0x0800012F
MOV R2,R0	0x08000134
B ENDL	0x08000138
MUL R2,...	0x0800013B
MUL R3,...	0x0800013F
ADD R2,R3	0x08000144
MOV R0,R2	0x08000146
BX LR	0x0800014A

Example: R2 = R0*R0+R1*R1

```
        MOV R0,#3
        MOV R1,#4
        BL  SSQ
        MOV R2,R0
        B  ENDL

SSQ     MUL R2,R0,R0
        MUL R3,R1,R1
        ADD R2,R2,R3
        MOV R0,R2
        BX  LR
ENDL    ...
```



R0	3
R1	4
R2	9
R3	16

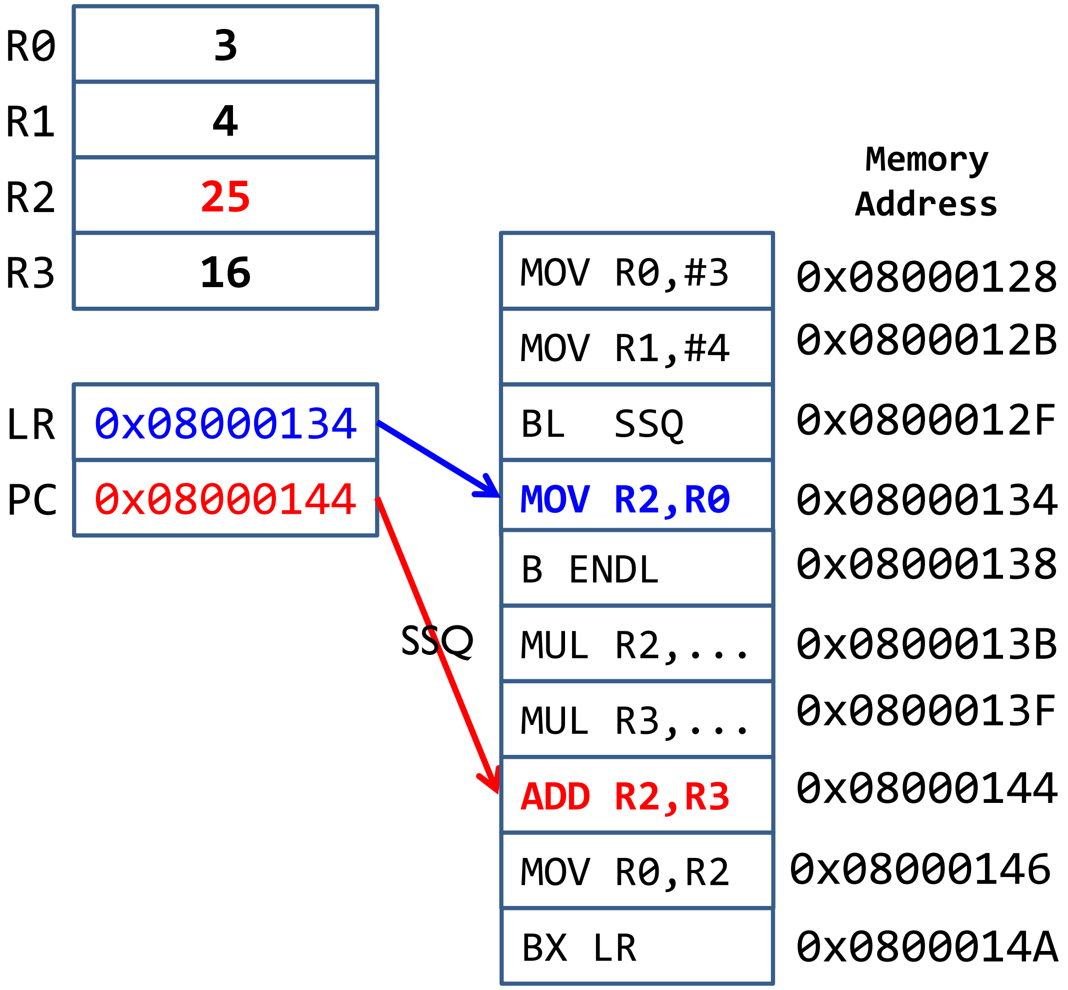
LR	0x08000134
PC	0x0800013F

	Memory Address
MOV R0,#3	0x08000128
MOV R1,#4	0x0800012B
BL SSQ	0x0800012F
MOV R2,R0	0x08000134
B ENDL	0x08000138
MUL R2,...	0x0800013B
MUL R3,...	0x0800013F
ADD R2,R3	0x08000144
MOV R0,R2	0x08000146
BX LR	0x0800014A

Example: R2 = R0*R0+R1*R1

```
        MOV R0,#3
        MOV R1,#4
        BL  SSQ
        MOV R2,R0
        B ENDL

SSQ     MUL R2,R0,R0
        MUL R3,R1,R1
        ADD R2,R2,R3
        MOV R0,R2
        BX  LR
ENDL    ...
```



R0	3
R1	4
R2	25
R3	16

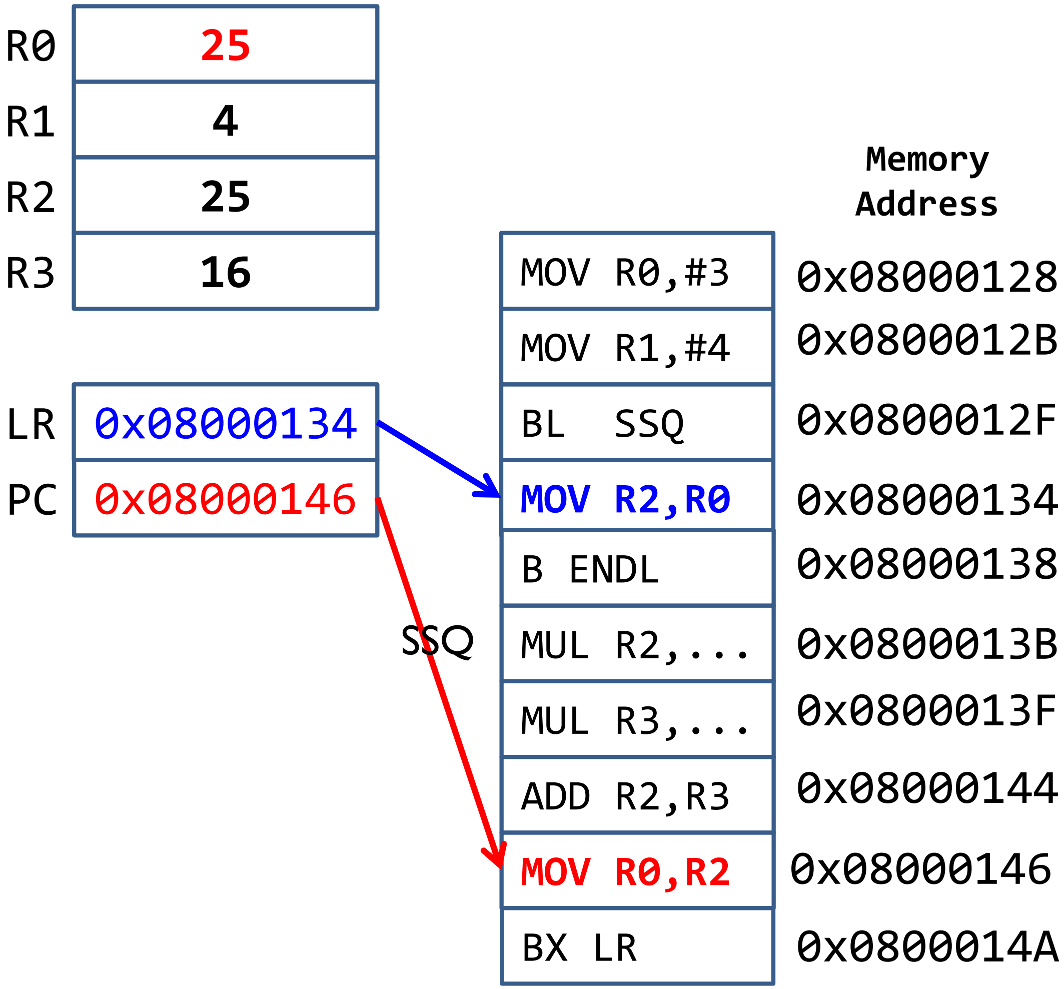
LR	0x08000134
PC	0x08000144

		Memory Address
	MOV R0,#3	0x08000128
	MOV R1,#4	0x0800012B
	BL SSQ	0x0800012F
	MOV R2,R0	0x08000134
	B ENDL	0x08000138
SSQ	MUL R2,...	0x0800013B
	MUL R3,...	0x0800013F
	ADD R2,R3	0x08000144
	MOV R0,R2	0x08000146
	BX LR	0x0800014A

Example: R2 = R0*R0+R1*R1

```
        MOV R0,#3
        MOV R1,#4
        BL  SSQ
        MOV R2,R0
        B  ENDL

SSQ     MUL R2,R0,R0
        MUL R3,R1,R1
        ADD R2,R2,R3
        MOV R0,R2
        BX  LR
ENDL    ...
```



R0	25
R1	4
R2	25
R3	16

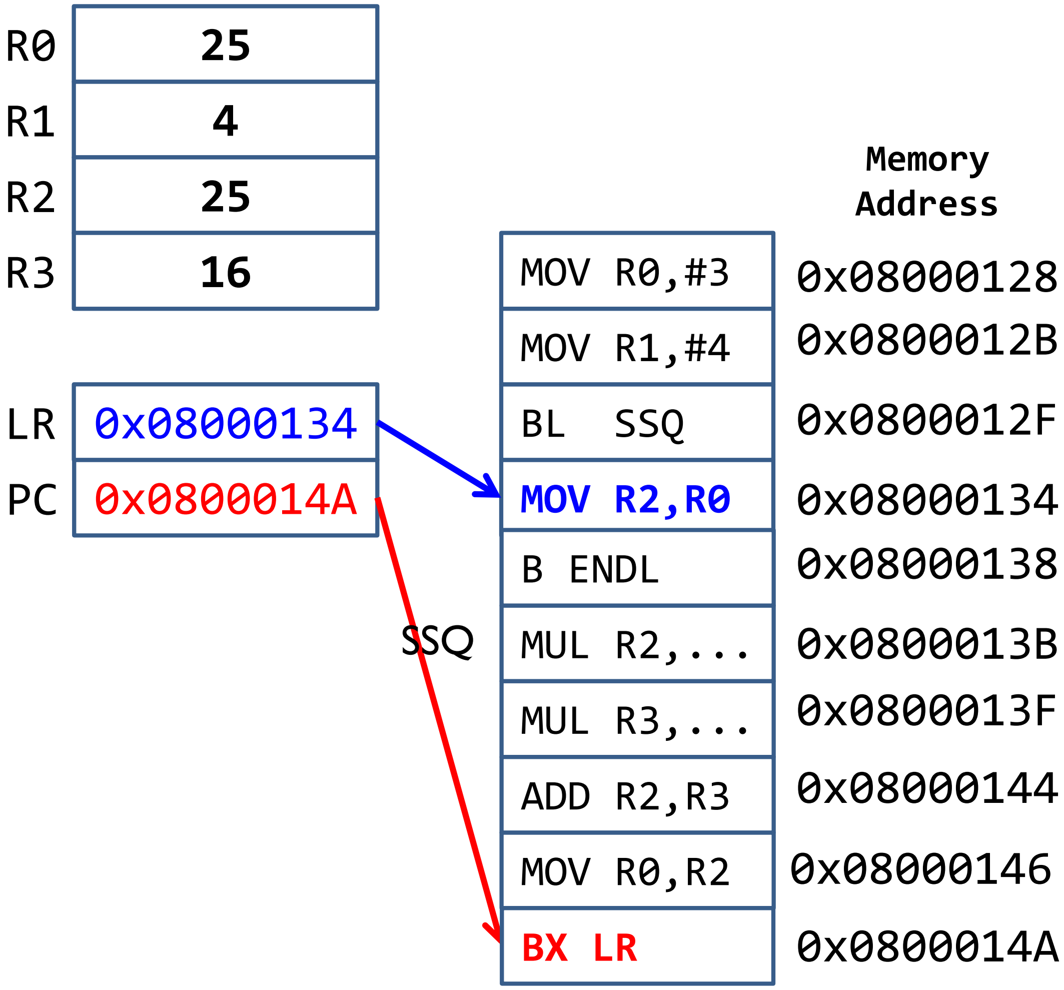
LR	0x08000134
PC	0x08000146

		Memory Address
	MOV R0,#3	0x08000128
	MOV R1,#4	0x0800012B
	BL SSQ	0x0800012F
	MOV R2,R0	0x08000134
	B ENDL	0x08000138
SSQ	MUL R2,...	0x0800013B
	MUL R3,...	0x0800013F
	ADD R2,R3	0x08000144
	MOV R0,R2	0x08000146
	BX LR	0x0800014A

Example: R2 = R0*R0+R1*R1

```
        MOV R0,#3
        MOV R1,#4
        BL  SSQ
        MOV R2,R0
        B ENDL

SSQ     MUL R2,R0,R0
        MUL R3,R1,R1
        ADD R2,R2,R3
        MOV R0,R2
        BX  LR
ENDL    ...
```



R0	25
R1	4
R2	25
R3	16

LR	0x08000134
PC	0x0800014A

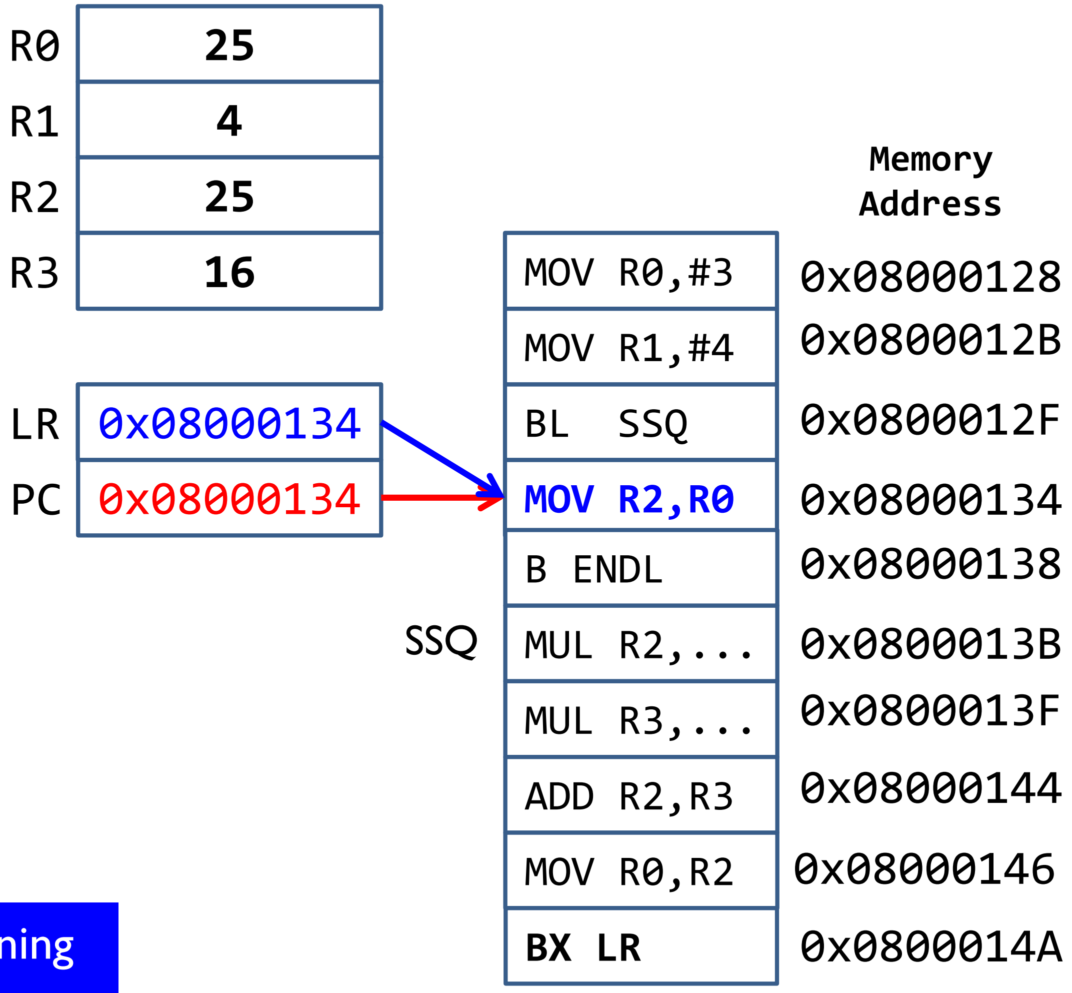
		Memory Address
	MOV R0,#3	0x08000128
	MOV R1,#4	0x0800012B
	BL SSQ	0x0800012F
	MOV R2,R0	0x08000134
	B ENDL	0x08000138
SSQ	MUL R2,...	0x0800013B
	MUL R3,...	0x0800013F
	ADD R2,R3	0x08000144
	MOV R0,R2	0x08000146
	BX LR	0x0800014A

Example: R2 = R0*R0+R1*R1

```
        MOV R0,#3
        MOV R1,#4
        BL  SSQ
        MOV R2,R0
        B ENDL

SSQ     MUL R2,R0,R0
        MUL R3,R1,R1
        ADD R2,R2,R3
        MOV R0,R2
        BX  LR
ENDL    ...
```

Copy LR to PC when returning from a subroutine!



R0	25
R1	4
R2	25
R3	16

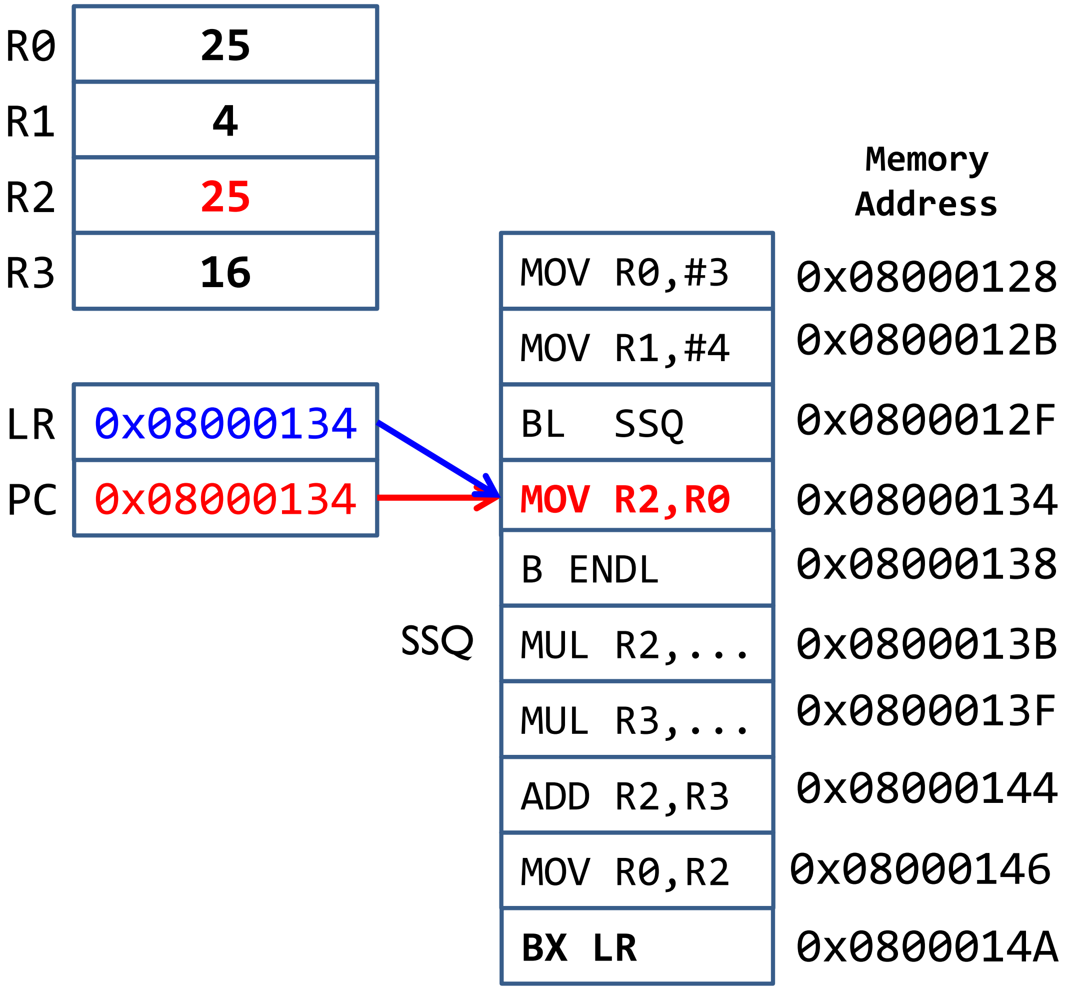
LR	0x08000134
PC	0x08000134

		Memory Address
	MOV R0,#3	0x08000128
	MOV R1,#4	0x0800012B
	BL SSQ	0x0800012F
	MOV R2,R0	0x08000134
	B ENDL	0x08000138
SSQ	MUL R2,...	0x0800013B
	MUL R3,...	0x0800013F
	ADD R2,R3	0x08000144
	MOV R0,R2	0x08000146
	BX LR	0x0800014A

Example: R2 = R0*R0+R1*R1

```
        MOV R0,#3
        MOV R1,#4
        BL  SSQ
        MOV R2,R0
        B ENDL

SSQ     MUL R2,R0,R0
        MUL R3,R1,R1
        ADD R2,R2,R3
        MOV R0,R2
        BX  LR
ENDL    ...
```



R0	25
R1	4
R2	25
R3	16

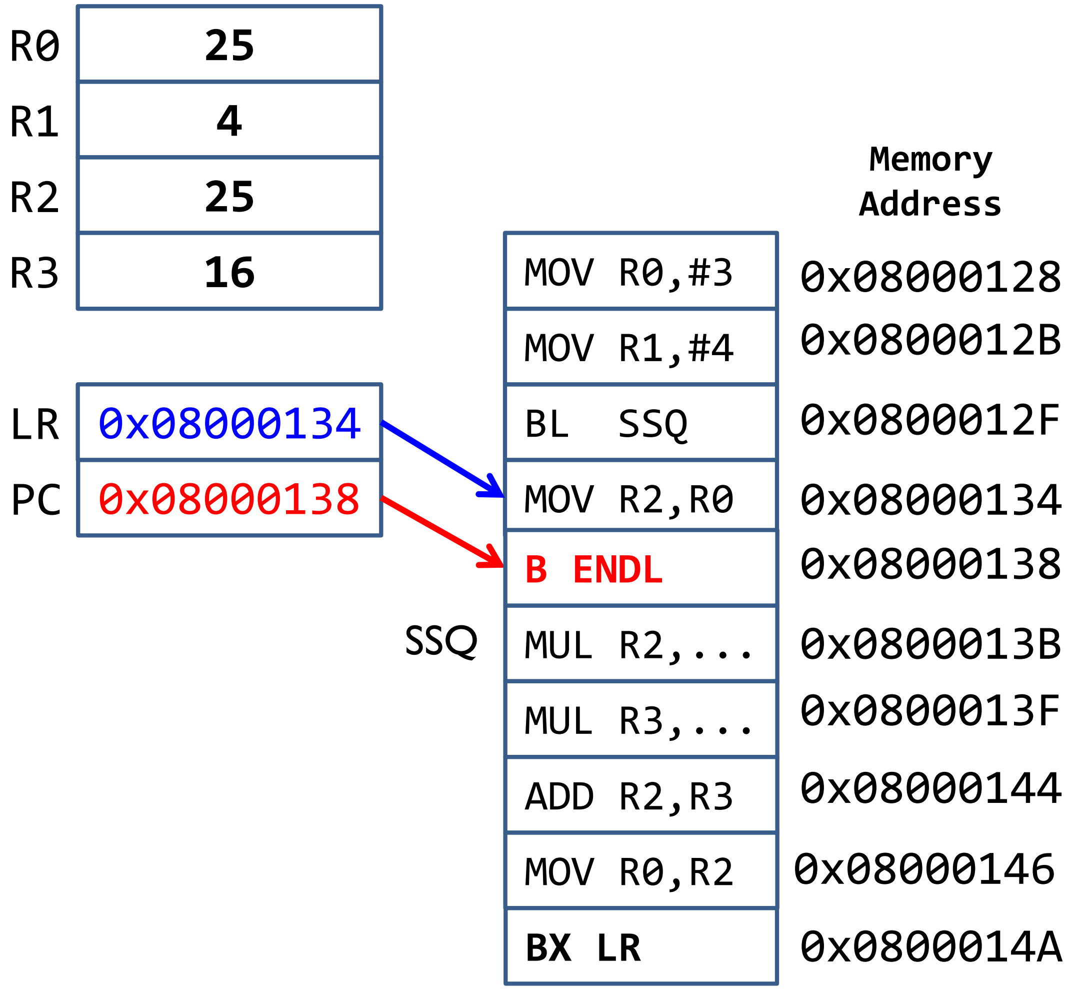
LR	0x08000134
PC	0x08000134

		Memory Address
	MOV R0,#3	0x08000128
	MOV R1,#4	0x0800012B
	BL SSQ	0x0800012F
	MOV R2,R0	0x08000134
	B ENDL	0x08000138
SSQ	MUL R2,...	0x0800013B
	MUL R3,...	0x0800013F
	ADD R2,R3	0x08000144
	MOV R0,R2	0x08000146
	BX LR	0x0800014A

Example: R2 = R0*R0+R1*R1

```
        MOV  R0,#3
        MOV  R1,#4
        BL   SSQ
        MOV  R2,R0
        B  ENDL

SSQ     MUL  R2,R0,R0
        MUL  R3,R1,R1
        ADD  R2,R2,R3
        MOV  R0,R2
        BX   LR
ENDL    ...
```



R0	25
R1	4
R2	25
R3	16

LR	0x08000134
PC	0x08000138

		Memory Address
	MOV R0,#3	0x08000128
	MOV R1,#4	0x0800012B
	BL SSQ	0x0800012F
	MOV R2,R0	0x08000134
	B ENDL	0x08000138
SSQ	MUL R2,...	0x0800013B
	MUL R3,...	0x0800013F
	ADD R2,R3	0x08000144
	MOV R0,R2	0x08000146
	BX LR	0x0800014A