Recursive Algorithm in AVR

Tim Dosen POK

Outline

- Recursive Addition
- Factorial
- Fibonacci

```
int Add(int m, int n) {
   if (n == 0) return m;
   else {
     int y = Add(m, n-1) + 1;
     return y;
   }
}
```

```
.def Temp=r16
     .def Num1=r5
     .def Num2=r6
     .def Rslt=r7
     Reset:
0000
      ldi
             Temp, low(RAMEND)
0001
             SPL,Temp
      out
0002
      ldi
             Temp, high(RAMEND)
0003
      out S
             PH,Temp
0004
      ldi
             Temp, 7
                              ; first number
0005
             Num1, Temp
      mov
      ldi
             Temp, 18
0006
                              ; second number
             Num2, Temp
0007
      mov
             recadd
      rcall
0008
     forever:
             forever
      rjmp
0009
```

SRAM	Address
FF	0x0000
FF	0x025F

```
.def Temp=r16
     .def Num1=r5
     .def Num2=r6
     .def Rslt=r7
                                                        SRAM
                                                                   Address
                                                        FF
                                                                   0x0000
     Reset:
0000
      ldi
             Temp, low(RAMEND)
0001
             SPL,Temp
      out
                                                        FF
0002
      ldi
             Temp, high(RAMEND)
                                                        FF
0003
      out S
             PH,Temp
                                                        FF
                              ; first number
      ldi
0004
             Temp, 7
                                                        FF
0005
             Num1, Temp
      mov
                                                        FF
      ldi
             Temp, 18
0006
                               ; second number
             Num2, Temp
                                                        FF
0007
      mov
             recadd
      rcall
0008
                                                        00
     forever:
                                                                   0x025F
                                                 push
                                                        09
             forever
      rjmp
0009
```

```
recadd:
int Add (int m, int n) {
                                                       tst Num2
/* base case */
                                                       brne notzero
  if (n == 0) return m;
                                                       mov Rslt, Num1
                                                       ret
/* recursive case */
                                              notzero:
     else {
                                                       dec Num2
         int y = Add (m, n-1) + 1;
                                                       rcall recadd
         return y;
                                                       inc Rslt
                                                       ret
```

tst

TST - Test for Zero or Minus

Description:

Tests if a register is zero or negative. Performs a logical AND between a register and itself. The register will remain unchanged.

Operation:

(i) $Rd \leftarrow Rd \bullet Rd$

Syntax: Operands:

Program Counter:

(i) TST Rd $0 \le d \le 31$

 $PC \leftarrow PC + 1$

16-bit Opcode: (see AND Rd, Rd)

0010	00dd	dddd	dddd

brne

BRNE – Branch if Not Equal

Description:

Conditional relative branch. Tests the Zero Flag (Z) and branches relatively to PC if Z is cleared. If the instruction is executed immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only if the unsigned or signed binary number represented in Rd was not equal to the unsigned or signed binary number represented in Rr. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 1,k).

Operation:

(i) If $Rd \neq Rr (Z = 0)$ then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Syntax: Operands: (i) BRNE k $-64 \le k \le +63$

Program Counter: PC ← PC + k + 1

PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k001

Status Register (SREG) and Boolean Formula:

I	Т	Н	S	V	N	Z	С
_	-	-	-	-	-	-	-

recadd:

000A tst Num2

000B **brne notzero**

000C mov Rslt, Num1

000D ret

notzero:

000E dec Num2

000F rcall recadd

0010 inc Rslt

0011 **ret**

SRAM	Address
FF	0x0000
•••	
FF	
00	
09	0x025F

recadd:

000A tst Num2

000B **brne notzero**

000C mov Rslt, Num1

000D ret

notzero:

000E dec Num2

000F rcall recadd

0010 inc Rslt

0011 **ret**

SRAM	Address
FF	0x0000
•••	
FF	
FF	
FF	
FF	
00	
10	0x025D
00	
09	0x025F

recadd: 000A tst Num2 000B brne notzero **Address SRAM** 000C mov Rslt, Num1 FF 0x0000000D ret notzero: FF dec Num2 000E FF rcall recadd 000F inc Rslt 0010 00 ret 0011 0x025B10

00

10

00

09

0x025D

0x025F

recadd:

000A tst Num2

000B **brne notzero**

000C mov Rslt, Num1

000D ret

notzero:

000E dec Num2

000F rcall recadd

0010 inc Rslt

0011 **ret**

... and so on, until Num2 is zero

SRAM	Address
FF	0x0000
•••	
00	
10	0x0259
00	
10	0x025B
00	
10	0x025D
00	
09	0x025F

	recadd:			
000A		tst Num2		
000B 000C		brne notzero mov Rslt, Num1	SRAM	Address
000D		ret	FF	0x0000
000E	notzero:	dec Num2	00	0x0228
000E 000F		rcall recadd	10	0x0229
0010		inc Rslt	00	
0011		ret	10	
			00	
	and so	on, until Num2 is zero		
			09	0x025F

recadd:

000A tst Num2

000B brne notzero

000C mov Rslt, Num1

000D ret

notzero:

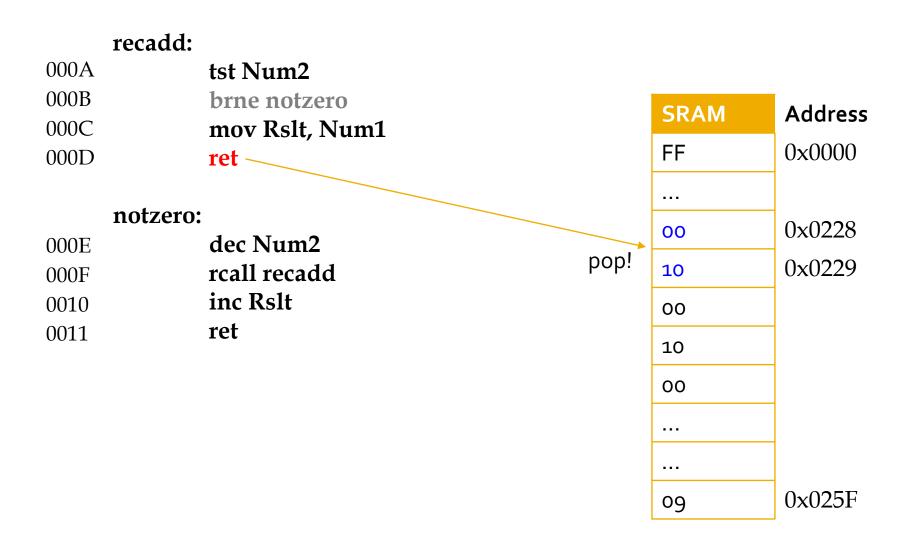
000E dec Num2

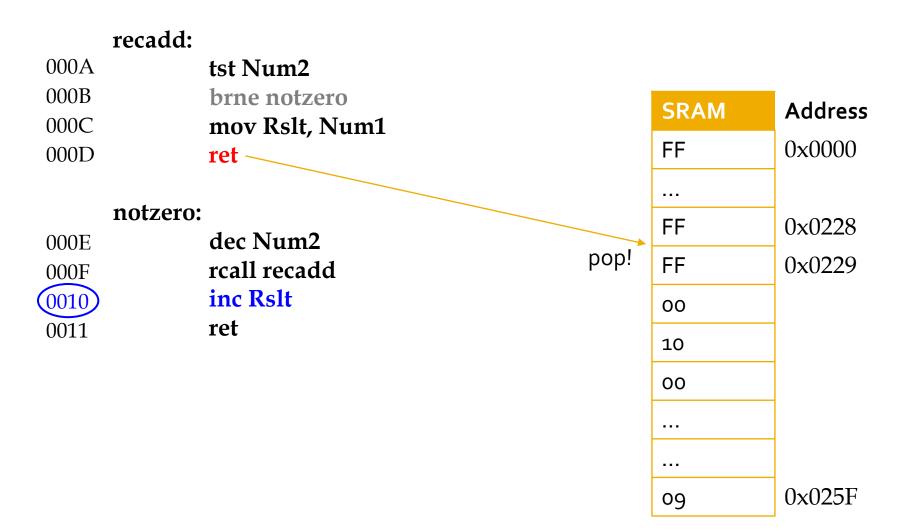
000F rcall recadd

0010 inc Rslt

0011 **ret**

SRAM	Address
FF	0x0000
00	0x0228
10	0x0229
00	
10	
00	
•••	
•••	
09	0x025F





000A	recadd:	tst Num2		
000B 000C		brne notzero mov Rslt, Num1	SRAM	Address
000D		ret	FF	0x0000
	notzero:			
000E	notzero.	dec Num2	FF	0x0228
000F		rcall recadd	FF	0x0229
0010		inc Rslt	00	
0011		retpop!	10	
			00	
			09	0x025F

	recadd:			
000A	tst Num	2		
000B 000C	brne not mov Rsl		SRAM	Address
000D	ret	y rumii	FF	0x0000
000E	notzero: dec Nun	12	FF	0x0228
000F	rcall reca		FF	0x0229
0010	inc Rslt		FF	
0011	ret	pop!	FF	
			00	
and so on		so on		
			09	0x025F

recadd:

000A tst Num2

000B **brne notzero**

000C mov Rslt, Num1

000D ret

notzero:

000E dec Num2

000F rcall recadd

0010 inc Rslt

0011 ret

SRAM	Address
FF	0x0000
FF	
00	
09	0x025F

recadd:

000A tst Num2

000B **brne notzero**

000C mov Rslt, Num1

000D ret

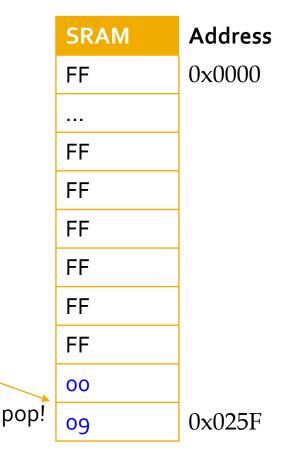
notzero:

000E dec Num2

000F rcall recadd

0010 inc Rslt

0011 **ret**



```
.def Temp=r16
     .def Num1=r5
     .def Num2=r6
     .def Rslt=r7
     Reset:
0000
      ldi
             Temp, low(RAMEND)
0001
             SPL,Temp
      out
0002
      ldi
             Temp, high(RAMEND)
0003
      out S
             PH,Temp
0004
      ldi
             Temp, 7
                              ; first number
0005
             Num1, Temp
      mov
      ldi
             Temp, 18
0006
                              ; second number
             Num2, Temp
0007
      mov
             recadd
      rcall
0008
     forever:
             forever
      rjmp
```

SRAM	Address
FF	0x0000
FF	0x025F

```
int factorial(int n) {
  if (n == 0) return 1;
  else return (n * factorial(n - 1));
}
```

```
include "m8515def.inc"
     .def
             temp = r18
             tempin = r19
     .def
                              ; Define temporary variable
             tempout = r20
     .def
                                                        SRAM
                                                                  Address
                                                        FF
                                                                  0x0000
     start:
      ldi
             temp,low(RAMEND)
             SPL,temp
      out
                                                        FF
             temp,high(RAMEND)
     ldi
0002
                                                        FF
             SPH,temp
0003
      out
                                                        FF
0004
     ldi
             r16, 5
                                                        FF
0005
             tempin, r16
      mov
                                                        FF
     rcall
             fact
0006
     forever:
                                                        FF
             forever
      rjmp
                                                        FF
```

0000

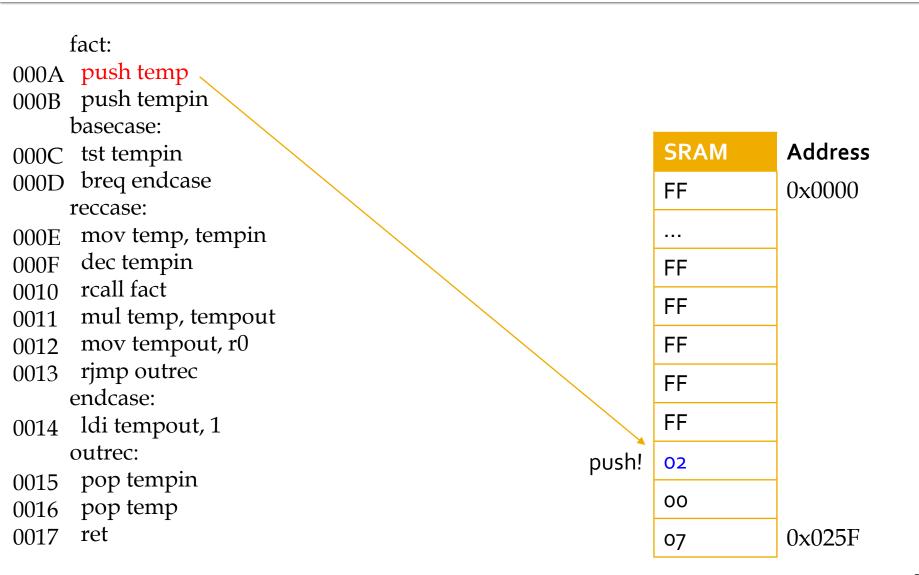
0001

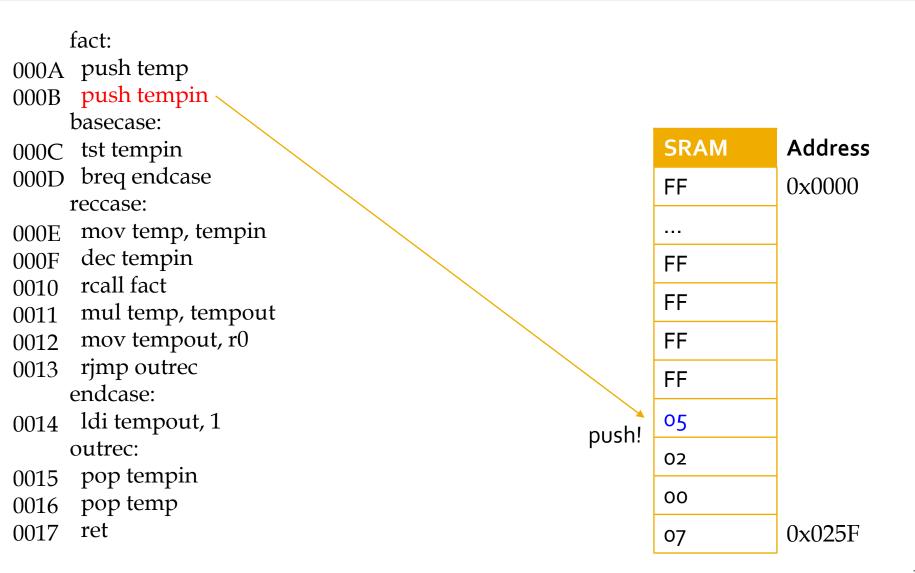
0007

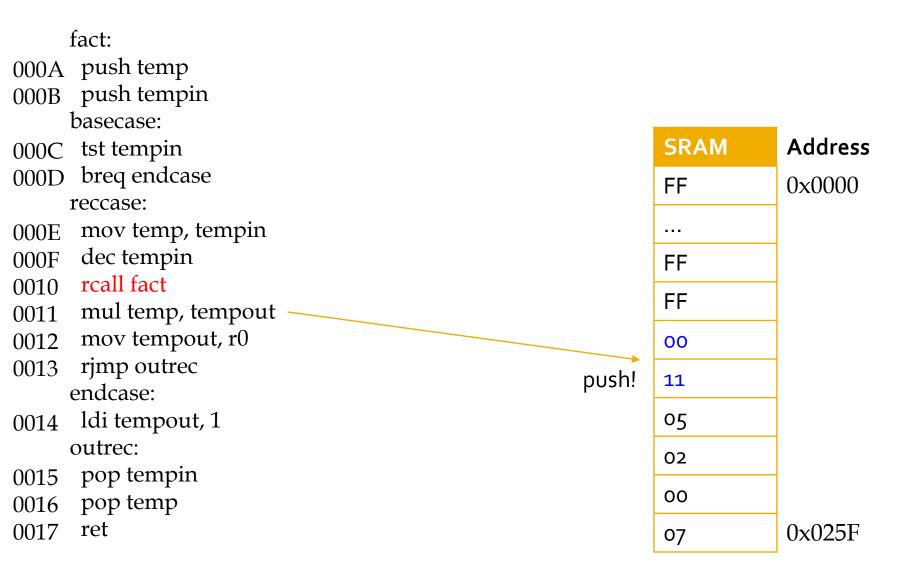
0x025F

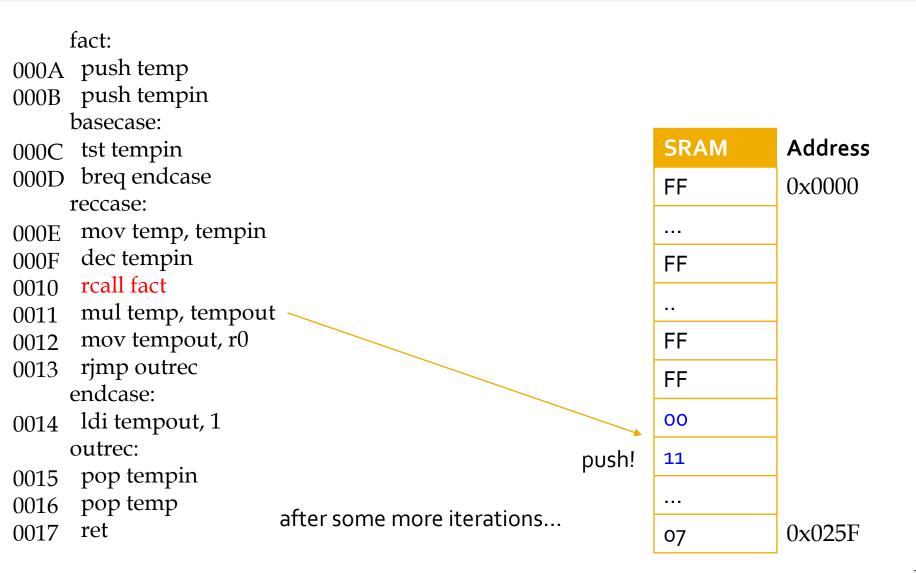
FF

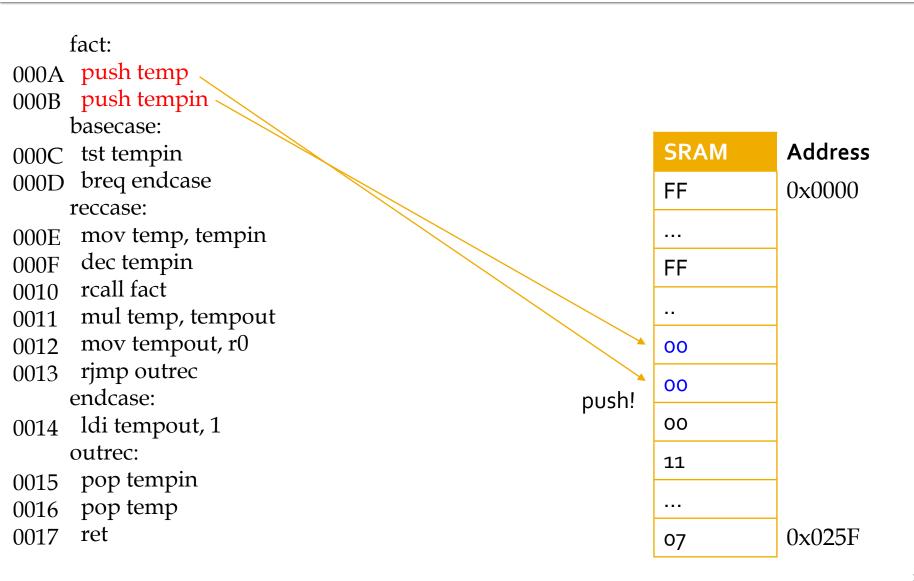
```
.include "m8515def.inc"
     .def
             temp = r18
             tempin = r19
     .def
                              ; Define temporary variable
             tempout = r20
     .def
                                                        SRAM
                                                                   Address
                                                        FF
                                                                   0x0000
     start:
      ldi
0000
             temp,low(RAMEND)
0001
             SPL,temp
      out
                                                        FF
             temp,high(RAMEND)
      ldi
0002
                                                        FF
             SPH,temp
0003
      out
                                                        FF
0004
      ldi
             r16, 5
                                                        FF
0005
             tempin, r16
      mov
                                                        FF
     rcall
             fact
0006
     forever:
                                                        FF
      rjmp
             forever
0007
                                                        00
                                                  push!
                                                                   0x025F
                                                        07
```

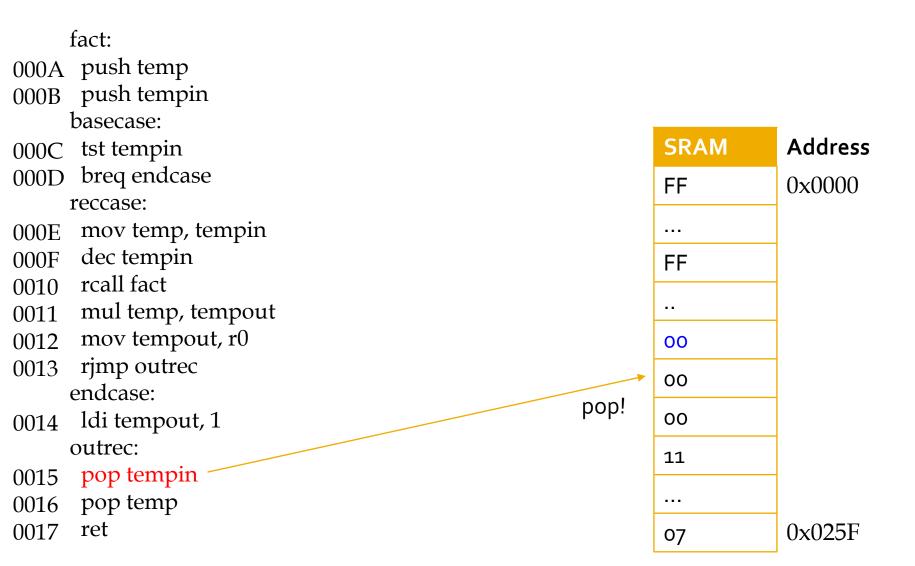


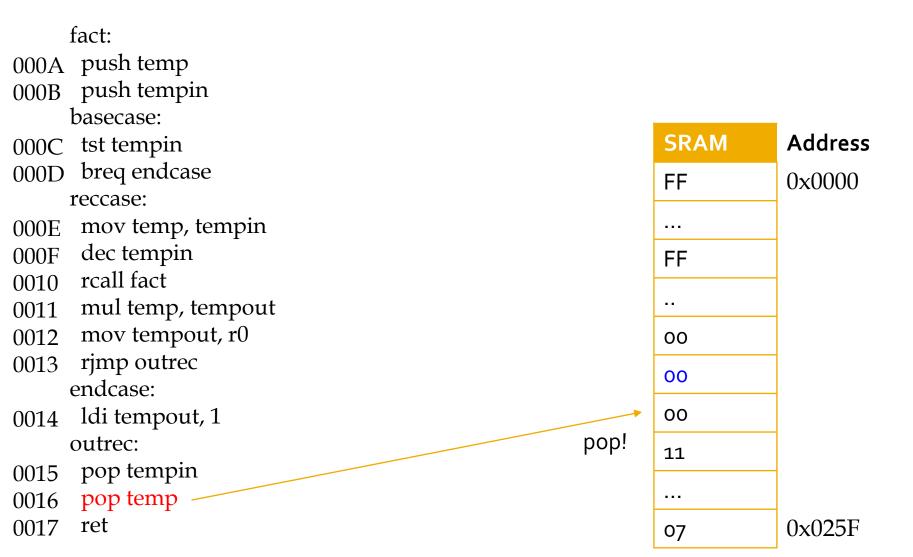


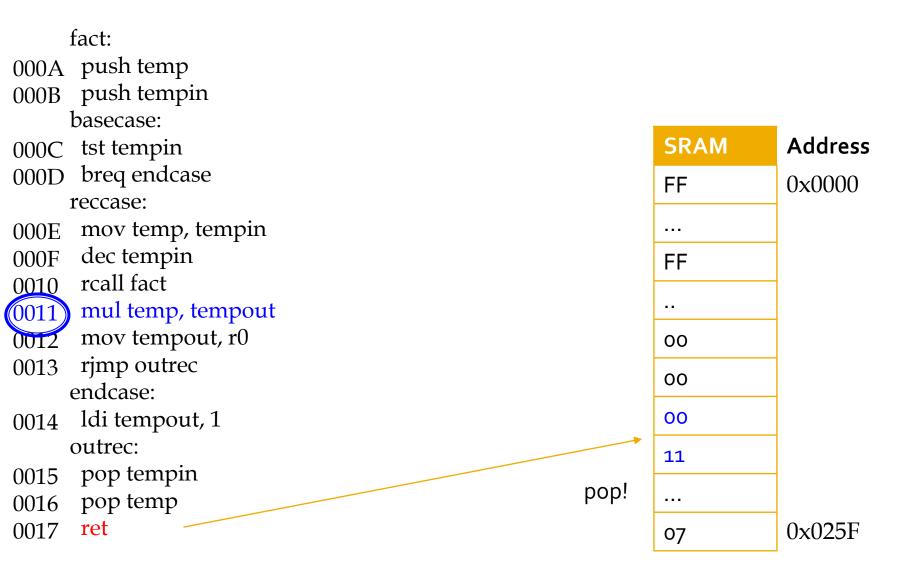












fact:		
000A push temp		
000B push tempin		
basecase:		Ī
000C tst tempin	SRAM	Address
000D breq endcase	FF	0x0000
reccase:	' '	07.0000
000E mov temp, tempin		
000F dec tempin		
0010 rcall fact		_
0011 mul temp, tempout	•••	-
0012 mov tempout, r0		
0013 rjmp outrec		-
endcase:	•••	-
0014 ldi tempout, 1 after some more iterations	05	
outrec:	02	
0015 pop tempin		_
0016 pop temp	00	
0017 ret pop!	07	0x025F

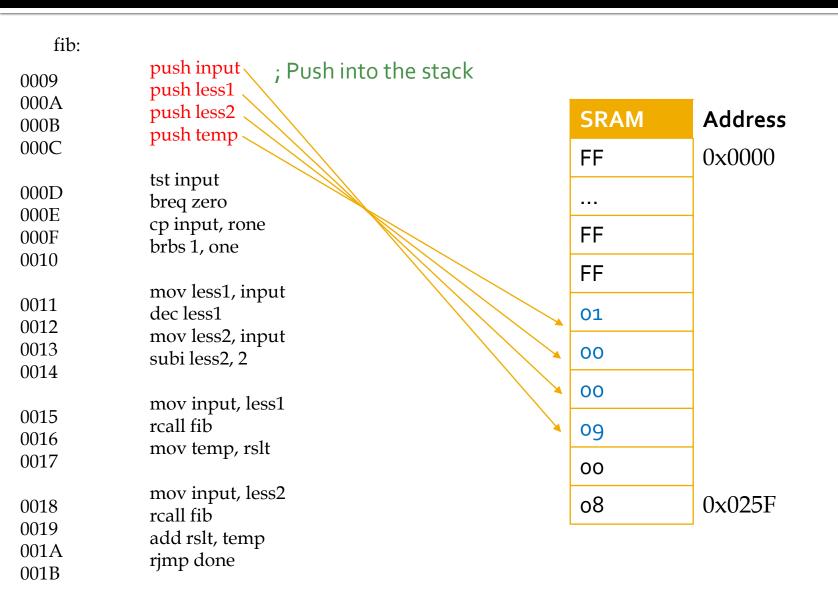
```
.include "m8515def.inc"
     .def
              temp = r18
              tempin = r19
     .def
                               ; Define temporary variable
              tempout = r20
     .def
                                                          SRAM
                                                                     Address
                                                          FF
                                                                     0x0000
     start:
      ldi
0000
              temp,low(RAMEND)
0001
              SPL,temp
      out
                                                          . . .
              temp,high(RAMEND)
      ldi
0002
              SPH,temp
0003
      out
                                                          ...
0004
      ldi
             r16, 5
                                                          . . .
0005
              tempin, r16
      mov
                                                          05
     rcall
              fact
0006
     forever:
                                                          02
      rjmp
              forever
                                                          00
```

0x025F

07

```
int fib(int n) {
  if (n <= 1) return n;
  else return (fib(n-1) + fib(n-2));
}</pre>
```

	.include "	'm8515def	.inc"			
	.def rone .def inpu	t = r16 = r17			SRAM	Address
	.def less1 = r4 .def less2 = r18			FF	0x0000	
	.def temp	r = r19				
0000	START:				FF	
0000 0001		ldi out	temp,low(RAMEND); Set stack po SPL,temp; last internal RAM lo		FF	
0002 0003		ldi	temp,high(RAMEND)		FF	
		out	SPH,temp		FF	
0004 0005		ldi temp, mov rone			FF	
		ldi input,	•		FF	
0006 0007	4	rcall fib			00	
0008	forever:	rjmp fore	ever	Push!	08	0x025F
0000		, 1				•



BRBS

BRBS - Branch if Bit in SREG is Set

Description:

(i)

Conditional relative branch. Tests a single bit in SREG and branches relatively to PC if the bit is set. This instruction branches relatively to PC in either direction (PC - $63 \le$ destination \le PC + 64). The parameter k is the offset from PC and is represented in two's complement form.

Operation:

BRBS s,k

(i) If SREG(s) = 1 then PC \leftarrow PC + k + 1, else PC \leftarrow PC + 1

Syntax: Operands:

 $0 \le s \le 7$, $-64 \le k \le +63$

Program Counter:

 $PC \leftarrow PC + k + 1$

PC ← PC + 1, if condition is false

16-bit Opcode:

Ì	1111	00kk	kkkk	ksss

Status Register (SREG) and Boolean Formula:

- 1	Т	Н	S	V	N	Z	С
_	-	-	-	-	-	-	-

76543210



CP – Compare

Description:

This instruction performs a compare between two registers Rd and Rr. None of the registers are changed. All conditional branches can be used after this instruction.

Operation:

(i) Rd - Rr

Syntax: Operands:

Program Counter:

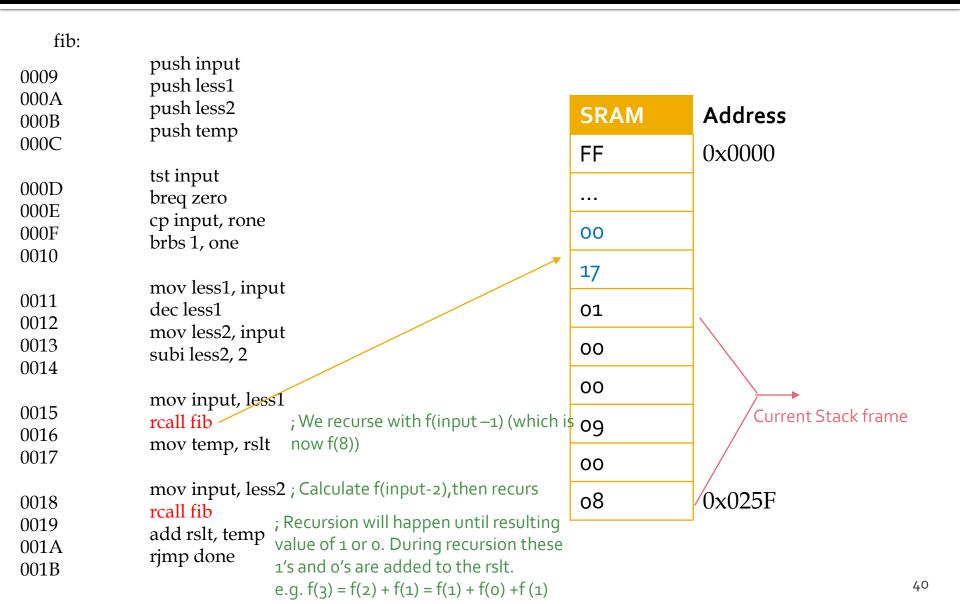
(i) CP Rd,Rr

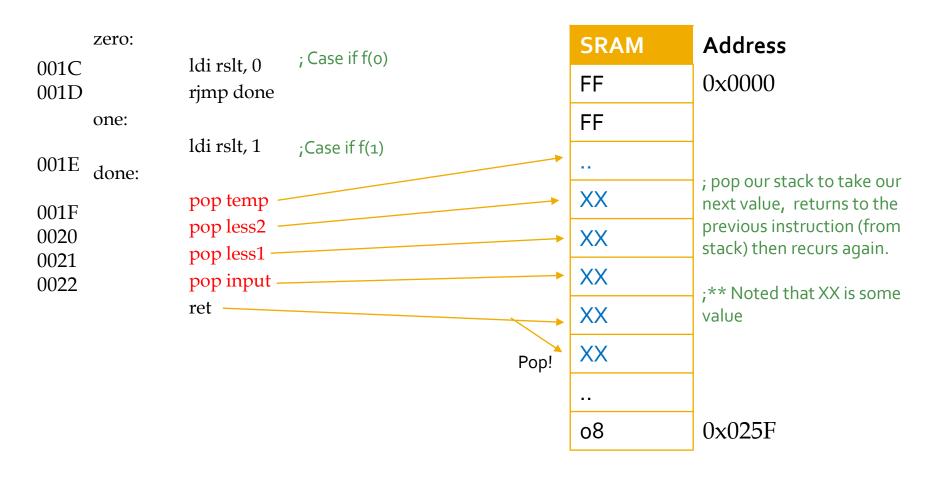
 $0 \le d \le 31, 0 \le r \le 31$

 $PC \leftarrow PC + 1$

16-bit Opcode:

0001	01rd	dddd	rrrr





zero: fib: ldi rslt, 0 push input rjmp done push less1 push less2 one: push temp ldi rslt, 1 done: tst input pop temp breq zero pop less2 cp input, rone brbs 1, one pop less1 pop input mov less1, input ret dec less1 mov less2, input subi less2, 2 int fib(int n) { if $(n \le 1)$ mov input, less1 rcall fib return n; mov temp, rslt else mov input, less2 return (fib(n-1) + fib(n-2));rcall fib add rslt, temp rjmp done