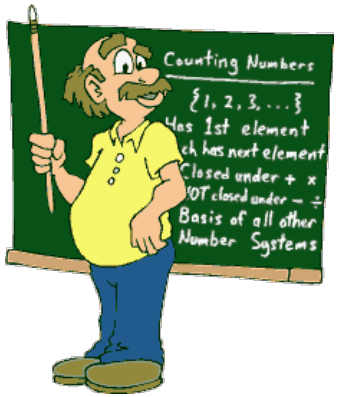
AARHUS
UNIVERSITY
SCHOOL OF ENGINEERING

MSYS

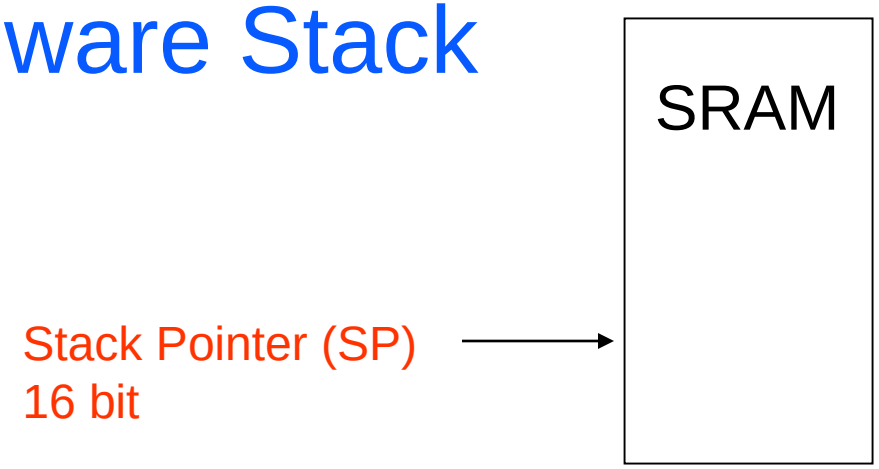
Microcontroller Systems

Lektion 7: Stack



Version: 12-9-2017, Henning Hargaard

Hardware Stack



- Stack pointer'en er et vigtigt register, som holder styr på, hvor "toppen af stacken" er (d.v.s. næste ledige plads (adresse)).
- Hver gang data skrives til stacken, flyttes SP automatisk **tilbage** (dekrementeres).
- Hver gang data hentes fra stacken, flyttes SP automatisk **frem** (inkrementeres).

Stack pointer (SP) = SPH og SPL

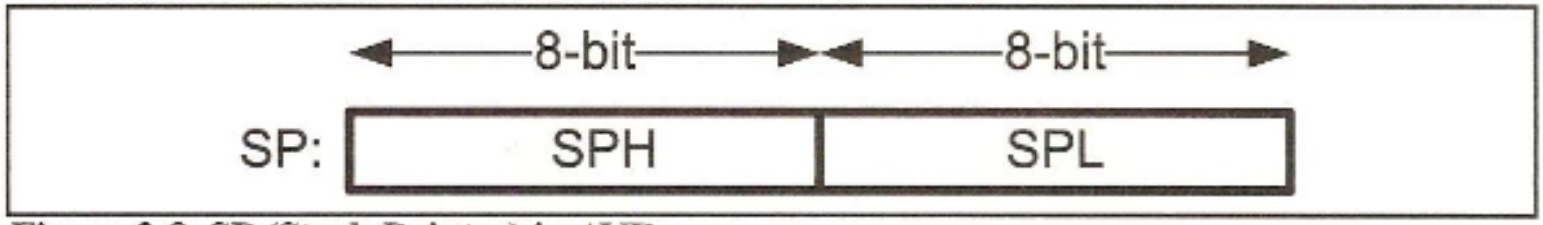


Figure 3-8. SP (Stack Pointer) in AVR

VIGTIGT:

Før brug skal stack pointeren initieres (sættes til at pege på ”toppen af stacken”) !

Det kan i Assembly gøres sådan:

```
LDI   R16, HIGH(RAMEND)
OUT   SPH,R16
LDI   R16, LOW(RAMEND)
OUT   SPL,R16
```

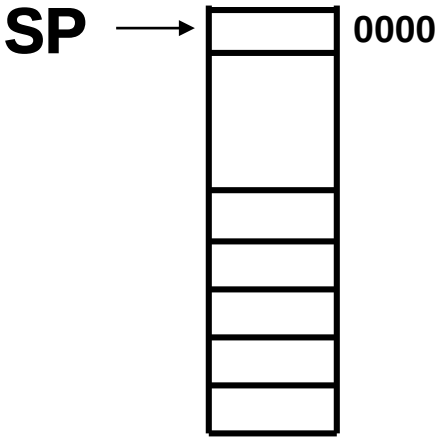
Mega32:
RAMEND = 0x085F
Mega2560:
RAMEND = 0x21FF

PUSH register / POP register

- Vi kan som (assembly-)programmører anvende stacken til at gemme data midlertidigt.
- ”PUSH register” (f.eks. PUSH R17) skriver register-indholdet til stacken (og **de**krementerer automatisk SP).
- ”POP register” (f.eks. POP R4) læser data fra stacken til registeret (og **in**krementerer automatisk SP).

Stack (PUSH og POP)

R20:	$10	R22:	$30
R21:	$20	R0:	$00



Address	Code
	ORG 0
0000	LDI R16,HIGH(RAMEND)
0001	OUT SPH,R16
0002	LDI R16,LOW(RAMEND)
0003	OUT SPL,R16
0004	LDI R20,0x10
0005	LDI R21,0x20
0006	LDI R22,0x30
0007	PUSH $10
0008	PUSH $20
0009	PUSH $30
000A	POP R21
000B	POP R0
000C	POP R20
000D	L1: RJMP L1

Test (”socrative.com”: Room = MSYS)

- Hvad indeholder R30 og R31 efter udførsel af følgende instruktioner:

```
LDI R30,30
LDI R31,31
PUSH R30
PUSH R31
POP R30
POP R31
```

A: R30 = 30 og R31 = 31
B: R30 = 30 og R31 = 30
C: R30 = 31 og R31 = 31
D: R30 = 31 og R31 = 30

CALL instruktionen

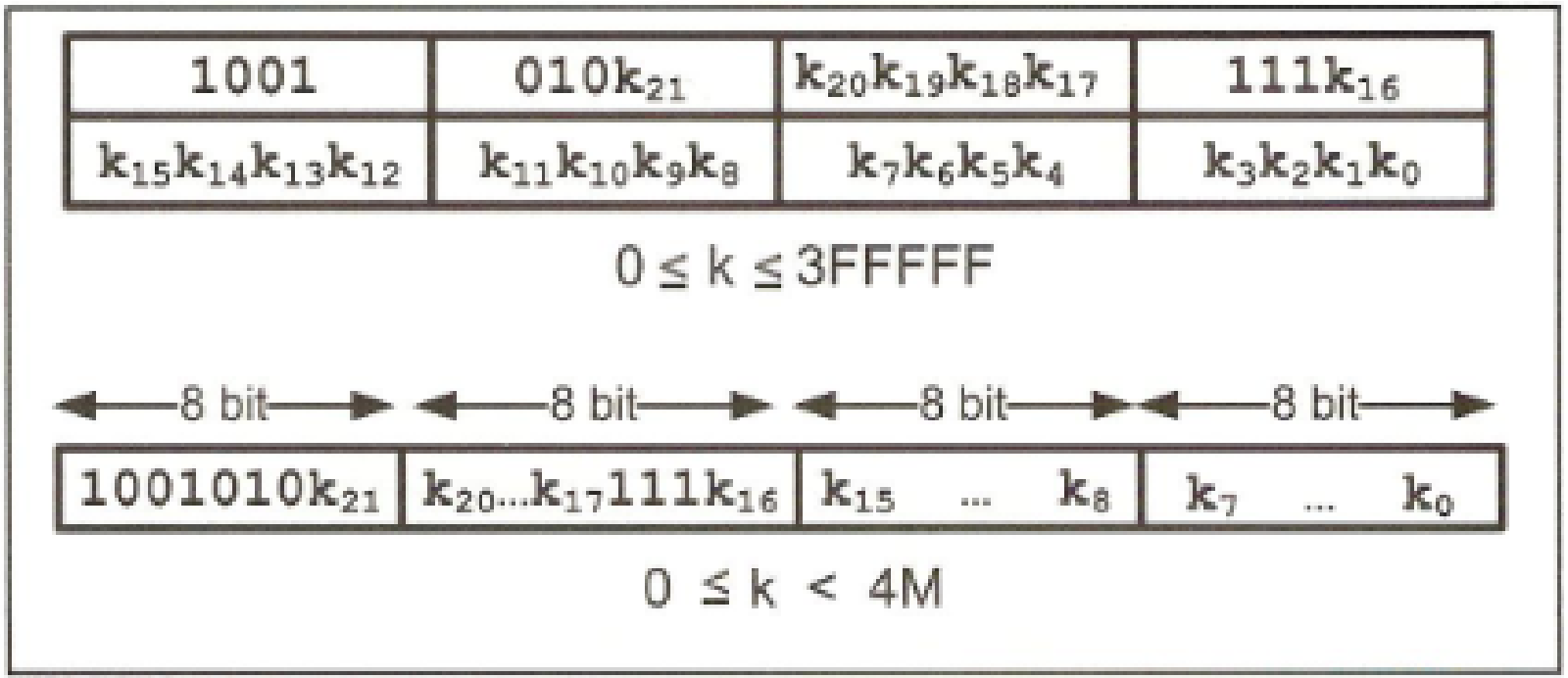
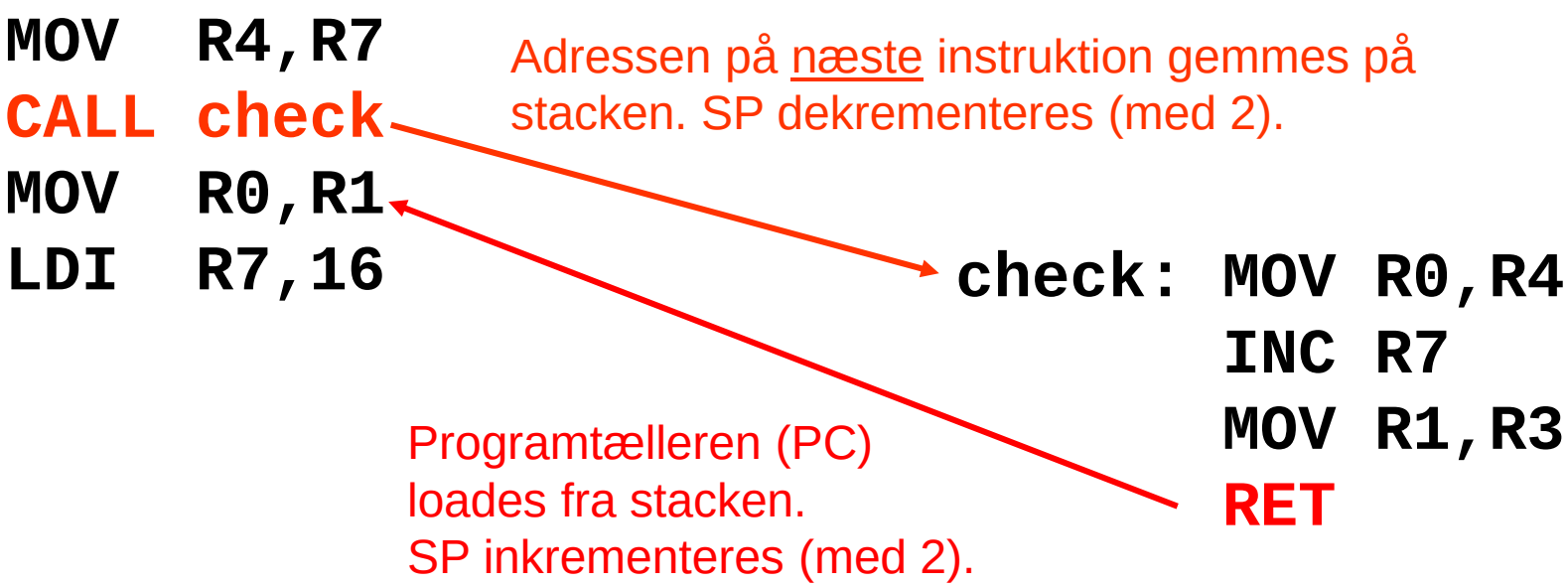


Figure 3-7. CALL Instruction Formation

Der findes også en ”relative CALL”, der fylder mindre.
Eksempel: RCALL HUGO

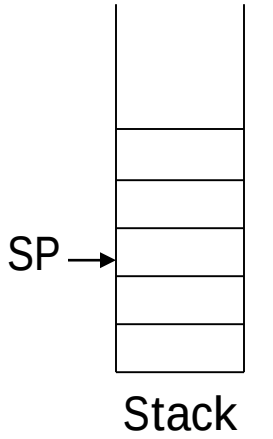
Subrutine-kald

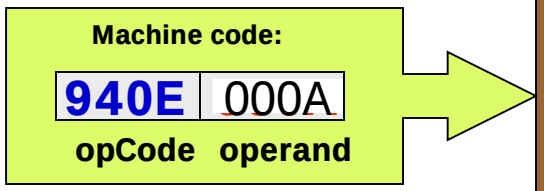


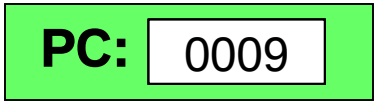
- Ved subrutinekald anvendes stacken (automatisk) til at ”finde tilbage” til instruktionen efter CALL :
  Der gemmes ”et bogmærke” på stacken.

Calling a Function

- To execute a call:
  - Address of the next instruction is saved
  - PC is loaded with the appropriate value

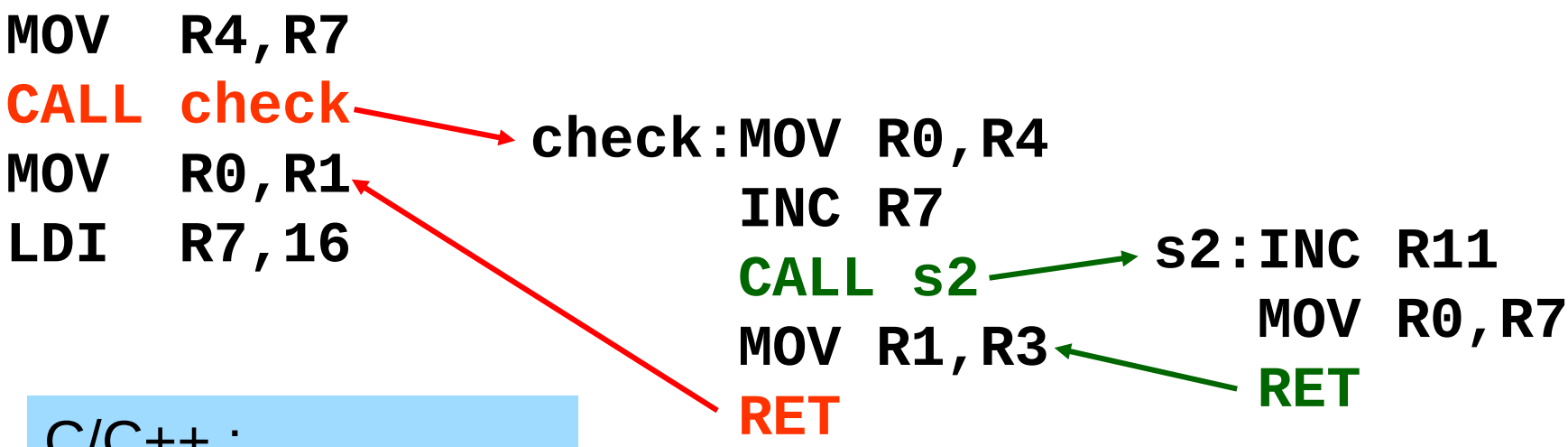






Address	Code
0000	LDI R16,HIGH(RAMEND)
0001	OUT SPH,R16
0002	LDI R16,LOW(RAMEND)
0003	OUT SPL,R16
0004	LDI R20,15
0005	LDI R21,5
0006	CALL FUNC_NAME
00 08	INC R20
0009	L1: RJMP L1
000A	FUNC_NAME:
000A	ADD R20,R21
000B	SUBI R20,3
000C	RET
000D	

Subrutine-kald i flere niveauer



C/C++ :
check(s2());

- Ved kald i flere niveauer holder SP hele tiden styr på, hvor den aktuelle retur-adresse er på stacken (nemlig ”toppen” af stacken).
- Pas på: Risiko for ”stack overflow” !

Typisk brug af CALL

```
.INCLUDE "M32DEF.INC"   ;Modify for your chip

;MAIN program calling subroutines
              .ORG   0
MAIN:         CALL   SUBR_1
              CALL   SUBR_2
              CALL   SUBR_3
              CALL   SUBR_4
HERE:         RJMP   HERE          ;stay here
;-----------end of MAIN
;
SUBR_1:       ....
              ....
              RET
;-----------end of subroutine 1
;
SUBR_2:       ....
              ....
              RET
;-----------end of subroutine 2

SUBR_3:       ....
              ....
              RET
;-----------end of subroutine 3

SUBR_4:       ....
              ....
              RET
;-----------end of subroutine 4
```

Figure 3-9. AVR Assembly Main Program That Calls Subroutines

Slut på lektion 7