Datorteknik

Tomas Nordström

Föreläsning 6

För utveckling av verksamhet, produkter och livskvalitet.

HÖGSKOLAN HALMSTAD

Föreläsning 6

- Vad händer vid uppstart
- SoC och Kringkretsar, PIO
- Programmering i Assembler
- Lab2 genomgång

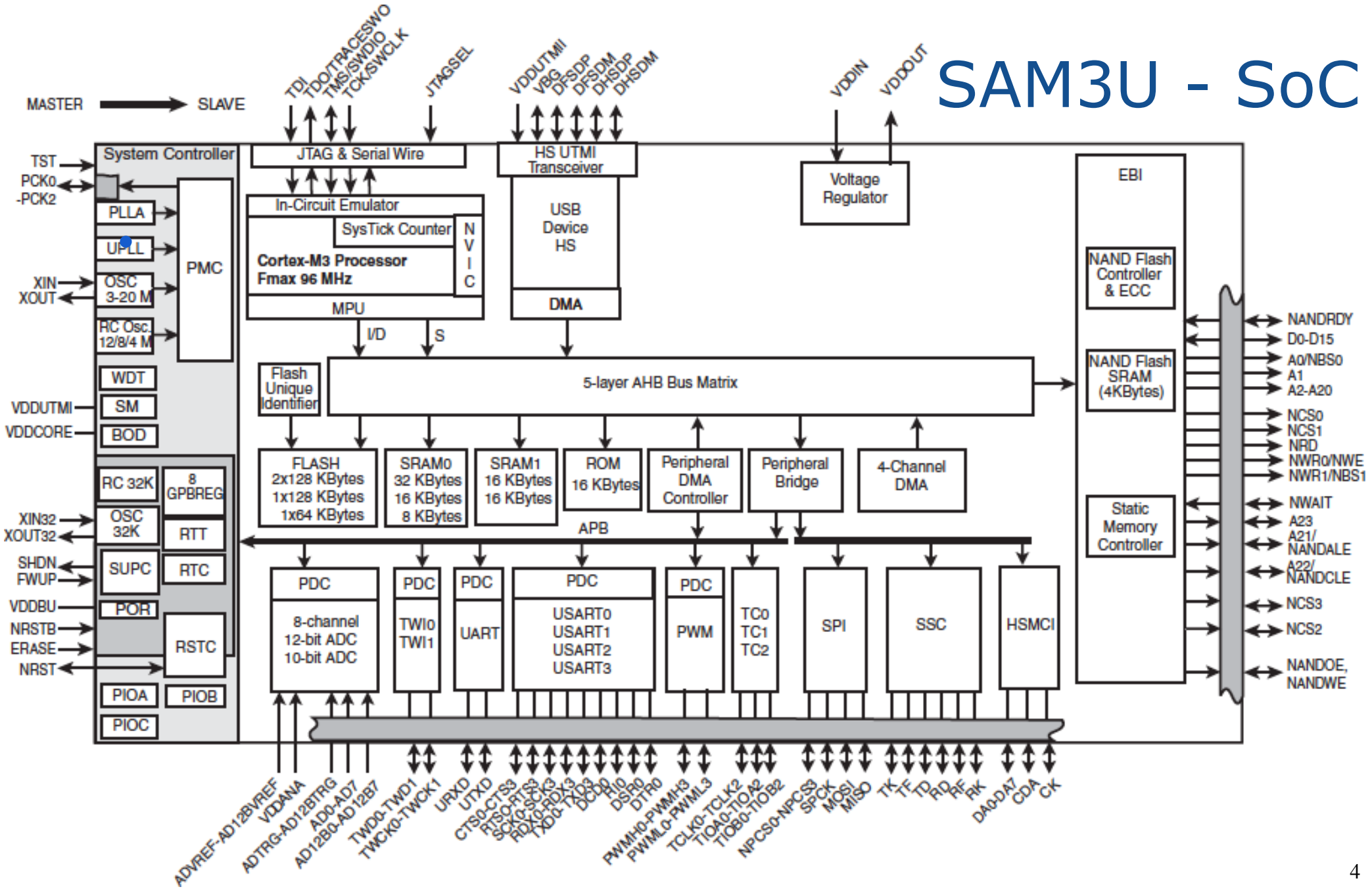
- References:
- [SUM3U-complete] ATMEL AT91SAM ARM-based Flash MCU - SAM3U Series (Complete manual), 2012

Vad händer vid start?

- Efter att man get kretsen ström eller reset så är processorn helt tom eller i okänt tillstånd. För att komma i känt tillstånd så görs
- Hämta ett ord/adress ifrån *minne*(0.x00000000) och lägg det i stackpekaren (SP=R14)
- Hämta ett ord/adress ifrån *minne*(0.x00000004) och lägg det i programräknaren (PC=R15), dvs hoppa till den adress som anges på adress 4.

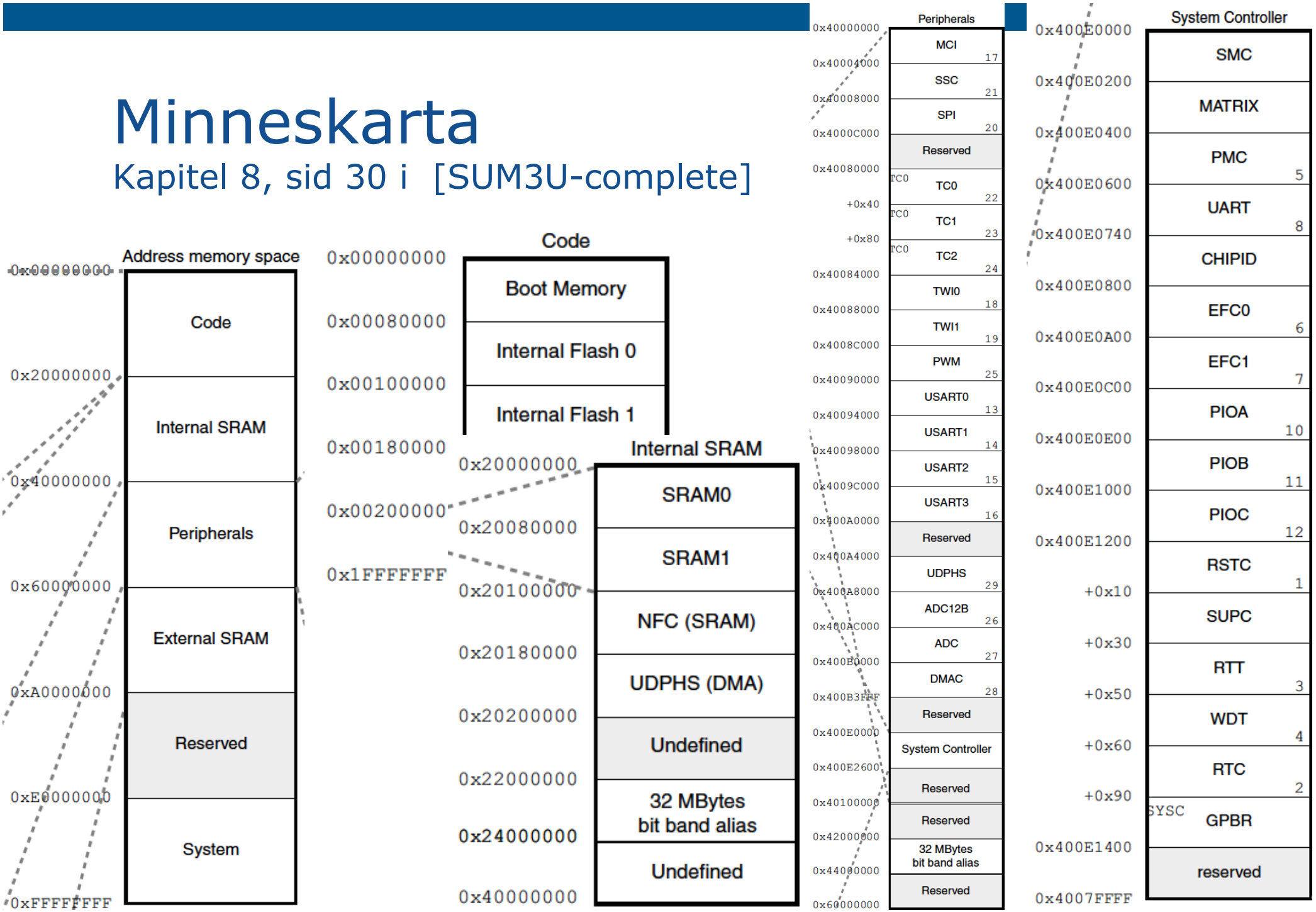
- Så efter reset måste ickeflyktigt minne ligga på adress 0 och framåt en bit.

SAM3U - SoC


MASTER
SLAVE
TDI
TDO/TRACESWO
TMS/SWDIO
TCK/SWCLK
JTAGSEL
VDDUTMII
VBG
DFSDP
DFSDM
DHSDP
DHSDM
VDDIN
VDDOUT
System Controller
JTAG & Serial Wire
HS UTMI Transceiver
Voltage Regulator
EBI
TST
PCK0
-PCK2
PLLA
UPLL
PMC
XIN
XOUT
OSC 3-20 M
RC Osc. 12/8/4 M
In-Circuit Emulator
SysTick Counter
NVIC
Cortex-M3 Processor
Fmax 96 MHz
MPU
USB Device HS
DMA
I/D
S
WDT
SM
BOD
VDDUTMI
VDDCORE
Flash Unique Identifier
5-layer AHB Bus Matrix
NAND Flash Controller & ECC
NAND Flash SRAM (4KBytes)
Static Memory Controller
RC 32K
8 GPBREG
XIN32
XOUT32
OSC 32K
RTT
SHDN
FWUP
SUPC
RTC
VDDBU
POR
NRSTB
ERASE
NRST
RSTC
PIOA
PIOB
PIOC
FLASH
2x128 KBytes
1x128 KBytes
1x64 KBytes
SRAM0
32 KBytes
16 KBytes
8 KBytes
SRAM1
16 KBytes
16 KBytes
ROM
16 KBytes
Peripheral DMA Controller
Peripheral Bridge
4-Channel DMA
APB
PDC
8-channel
12-bit ADC
10-bit ADC
TWI0
TWI1
UART
USART0
USART1
USART2
USART3
PWM
TC0
TC1
TC2
SPI
SSC
HSMCI
NANDRDY
D0-D15
A0/NBS0
A1
A2-A20
NCS0
NCS1
NRD
NWR0/NWE
NWR1/NBS1
NWAIT
A23
A21/ NANDALE
A22/ NANDCLE
NCS3
NCS2
NANDOE, NANDWE
ADVREF-AD12BVREF
VDDANA
ADTRG-AD12BTRG
AD0-AD7
AD12B0-AD12B7
TWD0-TWD1
TWCK0-TWCK1
URXD
UTXD
CTS0-CTS3
RTS0-RTS3
SCK0-SCK3
RDX0-RDX3
TXD0-TXD3
DCD0
RI0
DSR0
DTR0
PWMH0-PWMH3
PWML0-PWML3
TCLK0-TCLK2
TIOA0-TIOA2
TIOB0-TIOB2
NPCS0-NPCS3
SPCK
MOSI
MISO
TK
TF
TD
RD
RF
RK
DA0-DA7
CDA
CK

Minneskarta

Kapitel 8, sid 30 i [SUM3U-complete]



Hur mycket minne har vi?

- Se Kap 9, sid 31, i [SAM3U-Complete].
  - The SAM3U4 (256 KBytes internal Flash version) embeds a total of 48 Kbytes high-speed SRAM (32 Kbytes SRAM0 and 16 Kbytes SRAM1).
  - Enligt minneskartan startar SRAM0 på 0x20000000 och med 32kB (2^{15} byte) så kommer vi ha minne upp till 0x20007FFF.
  - Det är sedan ett hål tills SRAM1 startar!

Användning av minne

- Lägg kod i början av interna minnet
- Sätt stackpekaren till toppen av minnet (vi låter den växa neråt). SP måste vara på jämn ordgräns.

- I lab 3 så måste vi också ha plats för en avbrottsvektor, den läggs lämpligen före koden eller i den andra minnesbanken.

Watchdog Timer

- Kapitel 15 i [SUM3U-complete]:
  - The Watchdog Timer can be used to prevent system lock-up if the software becomes trapped in a deadlock.
  - It features a 12-bit down counter that allows a watchdog period of up to 16 seconds (slow clock at 32.768 kHz).
  - It can generate a general reset or a processor reset only. In addition, it can be stopped while the processor is in debug mode or idle mode.

- Slås automatiskt på efter reset
- Så om funktionen inte behövs så måste vi slå av den (inom 16 sekunder). Jmf sid 242 [SUM3U-complete]

Slå av watchdog timer

```
WDT_MR    EQU   0x400E1254 ; Watchdog Timer Mode Register

; disable WatchDog ---------------------------------
          LDR   R0,=WDT_MR
          LDR   R1,=0X8000
          STR   R1,[R0]
```

Power Management and Clocks

- Alla pereferienheter som inte används klockas inte för att spara ström.
- Så de enheter som vi vill använda måste klockas (sätt resp. bit i peripherial clock enable). Jmf sid 481 i [SUM3U-complete] och tabellen för “Peripheral Identifiers", på sid 43 i samma dockument.

```
; initiering Peripheral Clock ------------------------
PMC_PCER  EQU   0x400E0410 ; Peripheral Clock Enable Register

main      LDR   R0,=PMC_PCER
          LDR   R1,=0XC00
          STR   R1,[R0]
```

Exempelmall

```
#include "SAM3U.h"                    ; Fördeklarerade namn – fil som vi ska ta fram

Namn1               EQU <adress>
Namn2               EQU <tal>

                    THUMB
                    ORG <kodstart>

Main:               <sätt upp stacken>
                    <sätt upp avbrottsvektor>
                    <slå av watchdog timer>
                    <slå på peripherial clock för PortA&B>
                    <sätt upp portarna>
                    <övrig initiering>

                    <resten av koden>
Evigt               B Evigt

Konst               DCx      tal      ; konstanter x={8,16,32}
Utrymme             DSx      antal    ; reserverat utrymm
                    END
```

Övningsexempel: Subrutin Biträknare

- Skriv ett komplett program som räknar ettor i ett bitmönster. Gör en subrutin som tar hand om bit-räkningen. Talet antas vara ett 32-bitars tal.
- Bitmönstret har pushats på stacken innan anrop. Svaret ska läggas på samma position (så att det kan poppas av den anropande rutinen.
- Tips: Vänsterskiftning skiftar ut MSB i C-flaggan

Övningsexempel: Subrutin lowercase

- Konstruera ett program som gör om gemener till versaler. Endast gemener skall förändras!
- Anta att en teckensträng ligger i minnet med NULL (talet 0) som avslut. Adressen till strängen ligger på en adress kallad START.
- Tips: ASCII tecknen för gemenerna a-z ligger mellan 97-122 (0x61-0x7A)

Lab2

Målet med laborationen är att få begrepp om

- Subrutiner
  - Strukturering av en större kod med hjälp av subrutiner
  - Konstruktion av subrutiner
  - In/utparametrar.
  - Lokala variabler

- Maskning av bitar till ett register

Sammanfattning

- Bekanta dig med databladen!
- Med en bra mall så kan du fokusera på den viktiga koden. (Se till att ha en mall för uppstart och initiering; mall för huvudprogram; mall för subrutiner)