Load/store – več registrov

Z ukazom **ldm/stm (load multiple/store multiple)** je mogoče prebrati/shraniti več registrov:

- pomnilniški naslov za branje/shranjevanje mora biti **poravnan** (deljiv s 4)
- **registri z nižjimi indeksi** se vedno zapišejo na **nižji naslov**

Začetni naslov za shranjevanje/nalaganje je določen z baznim registrom in se pred ali po shranjevanju posameznega registra poveča ali zmanjša za 4. Pripona ukaza določa :

- ali se naj **naslov povečuje** ali **zmanjšuje**
- ali se to zgodi **pred ali po** branju/pisanju posameznega registra

Imamo štiri mogoče pripone:

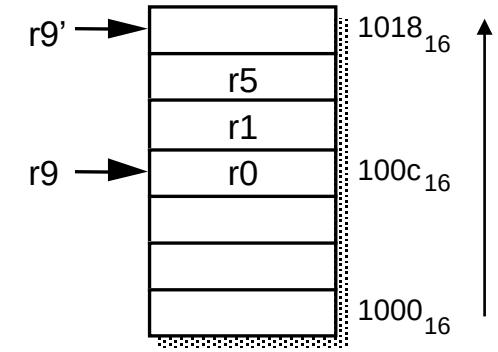
- **db** (decrement before), **da** (decrement after), **ib** (increment before), **ia** (increment after).

Če za baznim registrom stoji **!**, bo vrednost baznega registra enaka **naslovu po branju/shranjevanju zadnjega registra**. Sicer se vrednost baznega registra ne spremeni.

```
stmdb r13!, {r2-r9}      @ mem32[r13-4..r13-32] <- r9,…,r2
                         @ r13 <- r13-32   (8reg*4bajte=32bajtov)
stmdb r13, {r2-r9}       @ mem32[r13-4..r13-32] <- r9,…,r2
                         @ r13 ostane nespremenjen
ldmia r0!, {r2-r9}       @ r2,…,r9<-mem32[r0..r0+28]
                         @ r0 <- r0+32
stmda r1!, {r2-r9}       @ mem32[r1..r1-28] <- r9,…,r2
                         @ r1 <- r1-32
ldmib r13!, {r2-r9}      @ r2,…,r9 <- mem32[r13+4..r13+32]
                         @ r13 <- r13+32
```

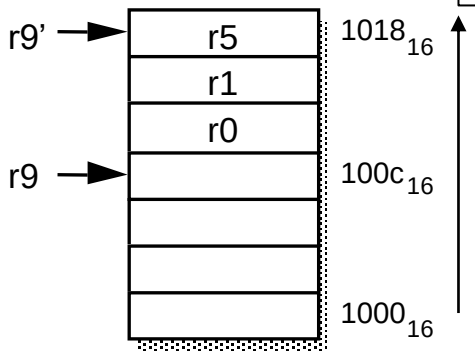
Load/store – več registrov

ia (inc. after)



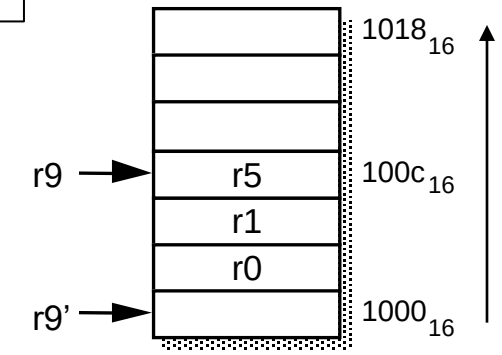

```
stmia r9!, {r0,r1,r5}
```

ib (inc. before)



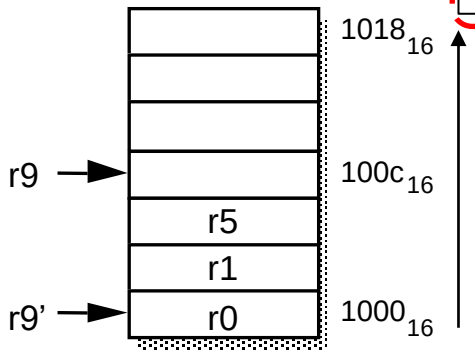

```
stmib r9!, {r0,r1,r5}
```

da (dec. after)




```
stmda r9!, {r0,r1,r5}
```

db (dec. before)




```
stmdb r9!, {r0,r1,r5}
```

Load/store – več registrov, bločno kopiranje vsebine

```
start
        LDR r0, =src            ; r0 = pointer to source block
        LDR r1, =dst            ; r1 = pointer to destination block

        MOV r2, #num            ; r2 = number of words to copy
        MOVS r3,r2, LSR #3      ; Number of eight word multiples
        …

octcopy LDM r0!, {r4-r11}       ; Load 8 words from the source
        STM r1!, {r4-r11}       ; and put them at the destination
        SUBS r3, r3, #1         ; Decrement the counter
        BNE octcopy             ; ... copy more
```

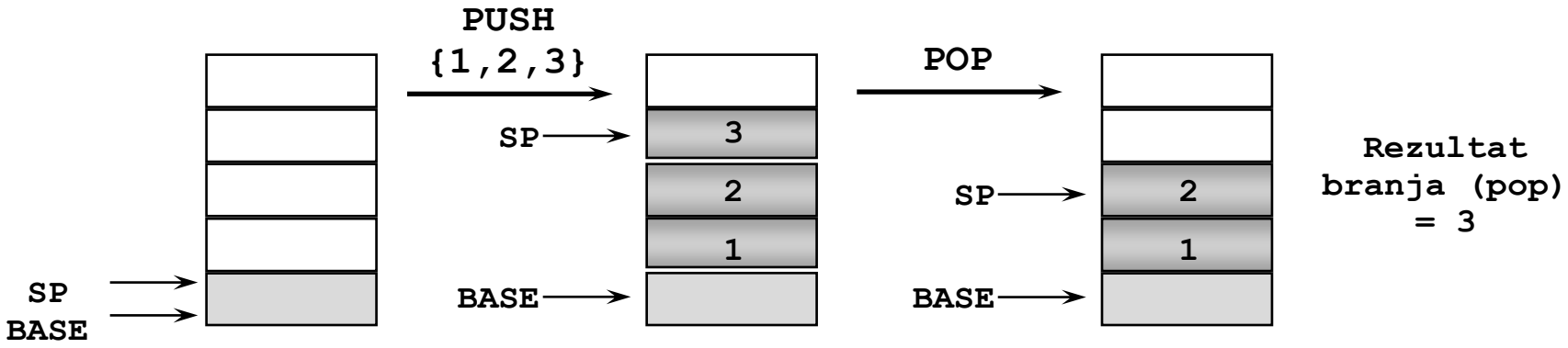
Sklad

Sklad je del pomnilnika, ki se:

- **poveča**, ko se operand shrani na **„vrh“** sklada - PUSH
- **zmanjša,** ko se podatek **prebere** iz vrha sklada - POP

Delovanje sklada zaznamujeta 2 kazalca :

- kazalec na začetni naslov („dno sklada“) - BASE
- **skladovni kazalec** („vrh sklada“) - SP - „Stack pointer“



Load/store – več registrov, sklad

Prenos **več registrov** se najpogosteje uporablja pri delu **s skladom** (shranjevanje na sklad, jemanje s sklada)

Podprte so vse različice skladov, od tod kratice:

- *ED (Empty Descending): širi se proti nižjim naslovom, SP kaže na prazen prostor*
- ***FD (Full Descending): širi se proti nižjim naslovom, SP kaže na zadnji element***
- *EA (Empty Ascending): širi se proti višjim naslovom, SP kaže na prazen prostor*
- *FA (Full Ascending): širi se proti višjim naslovom, SP kaže na zadnji element na skladu*

Uporabljamo FD sklad :

- ***vpis-DB: STMFD=STMDB***
- ***branje-IA: LDMFD=LDMIA***

		Ascending		Descending	
		Full	Empty	Full	Empty
Increment	Before	STMIB STMFA			LDMIB LDMED
	After		STMIA STMEA	LDMIA LDMFD	
Decrement	Before		LDMDB LDMEA	STMDB STMFD	
	After	LDMDA LDMFA			STMDA STMED

Podprogrami, sklad, uporaba/obnovitev registrov

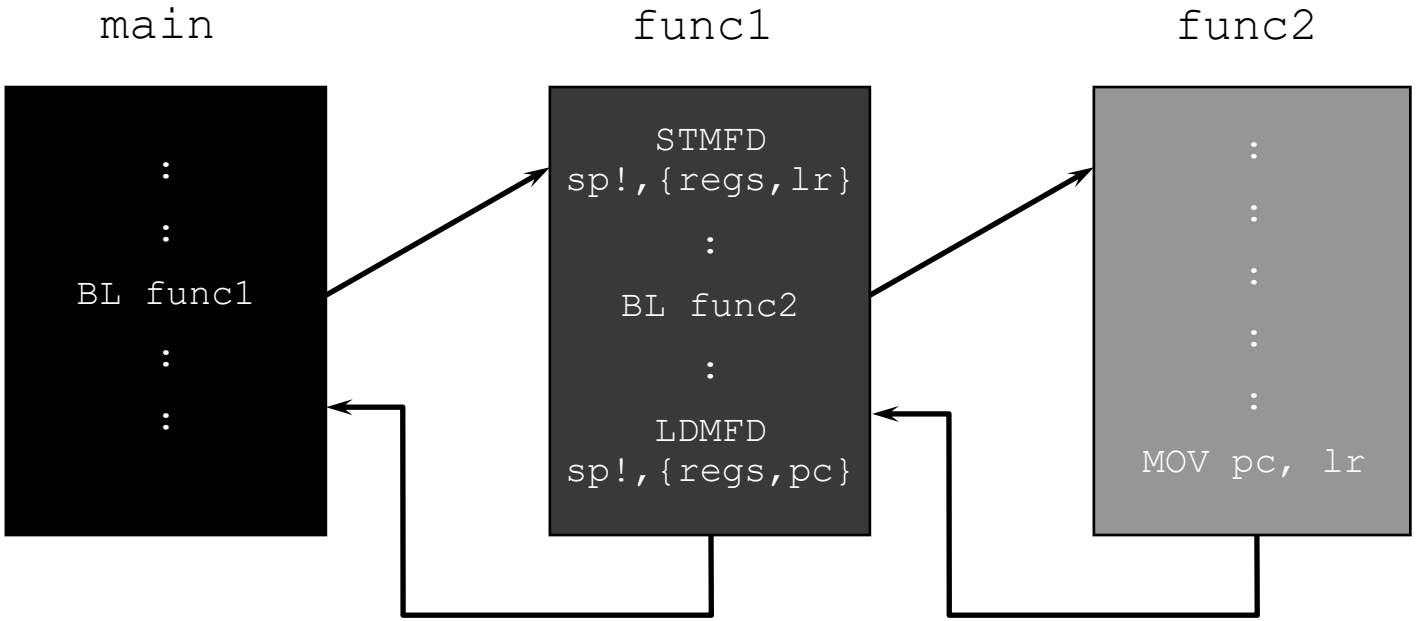
Kazalec na sklad je običajno register r13 (sp). Pred uporabo sklada moramo v r13 vpisati naslov vrha sklada. Pri določitvi tega naslova upoštevamo, da se sklad širi proti nižjim naslovom.

Podprogrami:

- ***klic podprograma:***
  - parametre v podprogram prenašamo v registrih od r0 naprej
  - na sklad se poleg "**delovnih registrov**" shrani tudi **r14** (lr - Link Register), v katerem je povratni naslov – s tem omogočimo **gnezdenje klicev podprogramov**
- ***vrnitev iz podprograma:***
  - "**delovni registri**" se obnovijo s sklada; povratni naslov se namesto v r14 zapiše v pc
- ***dogovor o rabi registrov:***
  - v podprogramu shranimo in obnovimo samo registre, ki so bili uporabljeni in niso služili za prenos parametrov – t.i. **delovni registri**

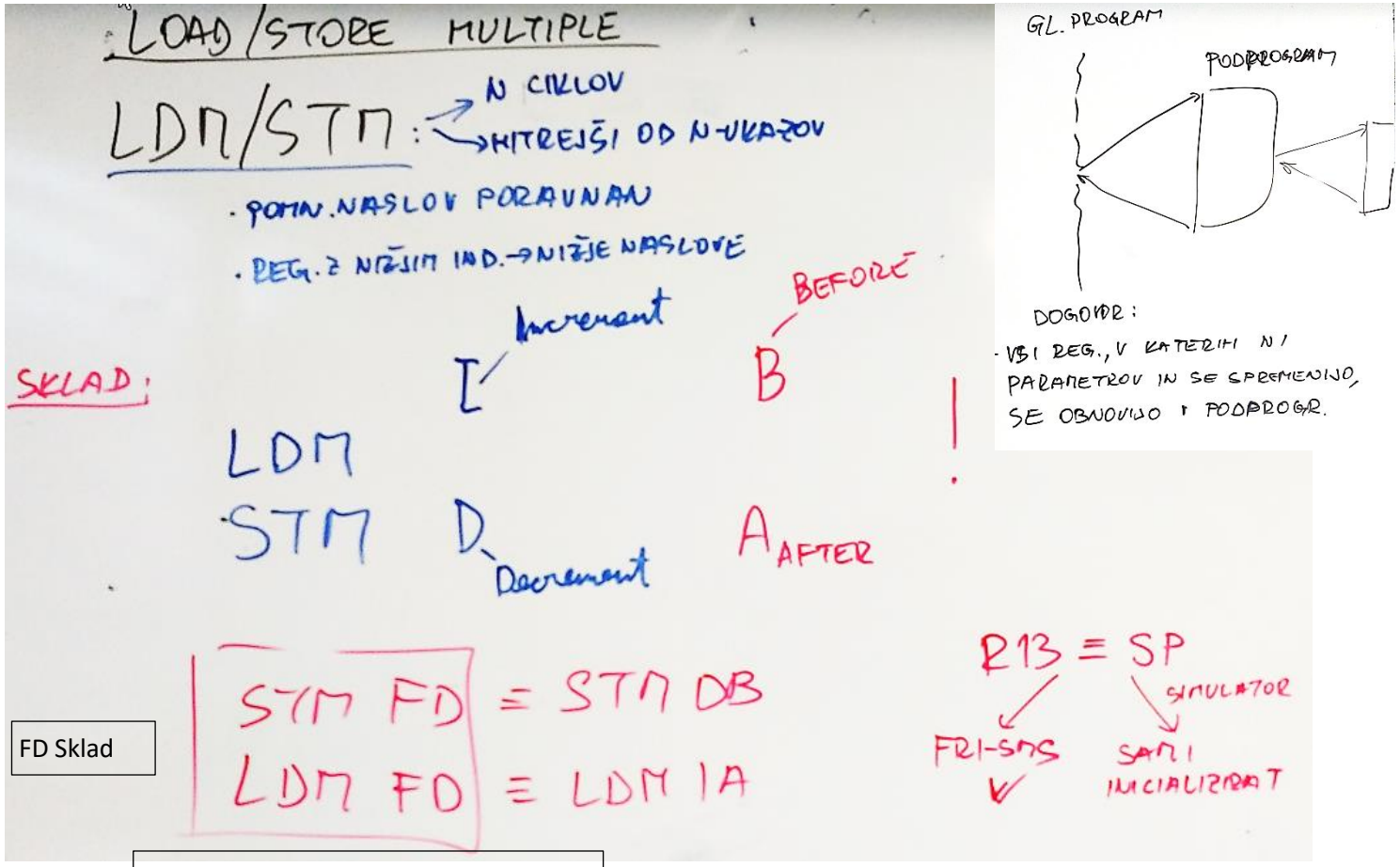
```
main:     ldr r13, =0x1000          @ initialize stack (stack pointer)
          mov r0, #10               @ put parameter in r0
          bl func1                  @ call subroutine func1
          …
-------------------------------------------------------------------------
func1:    stmfd r13!, {r1-r3,r14}   @ save work & link regs
          …                         @ inside sub1 we use regs r1,r2,r3
          bl func2                  @ call subroutine func2
          …
          …
          ldmfd r13!, {r1-r3,pc}    @ restore work regs & return
```

Podprogrami, sklad, uporaba/obnovitev registrov




```
main:     ldr r13, =0x1000          @ initialize stack (stack pointer)
          mov r0, #10               @ put parameter in r0
          bl func1                  @ call subroutine func1
          …
-------------------------------------------------------------------------
func1:    stmfd r13!, {r1-r3,r14}   @ save work & link regs
          …                         @ inside sub1 we use regs r1,r2,r3
          bl func2                  @ call subroutine func2
          …
          …
          ldmfd r13!, {r1-r3,pc}    @ restore work regs & return
```

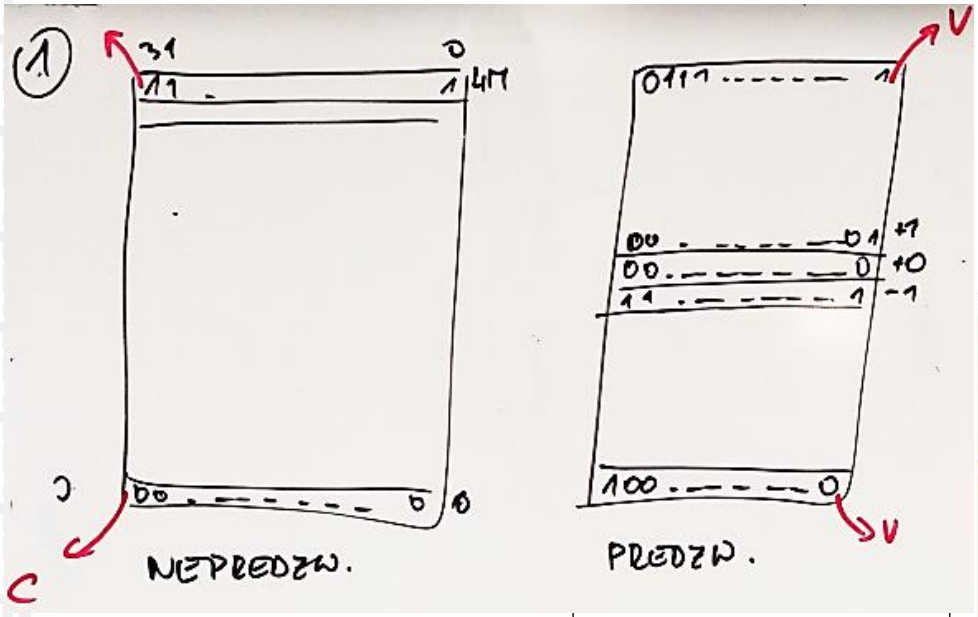
OR LAB - 3 : Tabla



FD Sklad

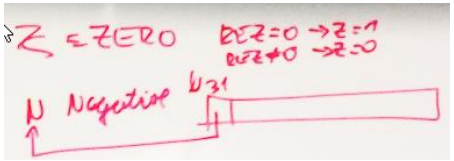
Standardni, pogosto uporabljen sklad.

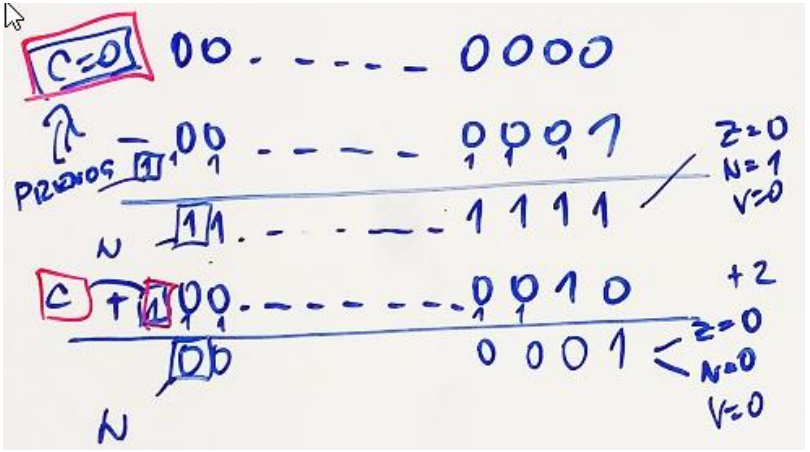
OR LAB - 3 : Tabla



Nepredznačena (0 .. +4milj.)

Predznačena (-2milj ..+2milj.)





CMP VEDNO VPLIVA NA ZAST

MOVS

ADD -

SUB -

Delo na STM32H7 razvojnem sistemu

Priključitev :

- **Mikro USB** priklop na **daljši stranici (nad LCD, srednji !!!)**

Poseben začetni projekt (github) in info za *STM32H7* (e-učilnica):

- ***dodajanje vsebine (Main.s):***

--------- Razvojni sistem STM32H750-DK ----------

- STM32H750B-DK Discovery kit with STM32H750XB MCU
- ORLab-STM32H7 - GitHub repozitorij
- User Manual Discovery kit stm32h750xb Uploaded 11/11/22, 10.15
- DataSheet_stm32h750xb Uploaded 11/11/22, 10.16
- Reference Manual rm0433-stm32h750xb Uploaded 11/11/22, 10.17
- Programming_Manual_pm0253-stm32h750xb Uploaded 11/11/22, 10.17
- Errata_es0396-stm32h750xb Uploaded 11/11/22, 10.19

CubeIDEWorkspace - stm32h7-asm/Core/Src/Main.s - STM32CubeIDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- CubeIDE_Workspace
  - stm32f4-asm-qemu
- Delo
  - ARM9Template
  - stm32f4-asm (in STM32AsmTemplate)
  - ARM9Template.zip
- Node_V4 (in node_v4)
- Sluzba
  - CAN_IEX_Module
  - ORLab-STM32H7
    - stm32h7-asm
      - Binaries
      - Includes
      - Core
        - Src
          - Main.s
        - Startup
          - startup_stm32h750xbhx.s
      - Debug
      - out
      - makefile
      - README.md
      - STM32H750X.svd
      - STM32H750XBHX_FLASH.ld
      - STM32H750XBHX_RAM.ld
    - README.md
  - RALab-STM32H7
    - stm32h7-asm_RA_LED
    - README.md
  - STM32_USB_Key_AdvDebug
  - STM32_USB_Key_FreeRTOS_AdvDebug
  - STM32CubeIDE_Adv_Debug
  - STM32F4_Discovery_VIN_Projects

Main.s × startup_stm32h750xbhx.s

```
12
13 ////////////////////////////////////////////////////////////////////////////
14 // Definitions
15 ////////////////////////////////////////////////////////////////////////////
16 // Definitions section. Define all the registers and
17 // constants here for code readability.
18
19 // Constants
20
21
22 // Start of data section
23         .data
24
25         .align
26
27 STEV1:  .word   0x10    // 32-bitna spr.
28 STEV2:  .word   0x40    // 32-bitna spr.
29 VSOTA:  .word   0       // 32-bitna spr.
30
31
32 // Start of text section
33         .text
34
35         .type  main, %function
36         .global main
37
38         .align
39 main:
40         ldr r0, =STEV1   // Naslov od STEV1 -> r0
41         ldr r1, [r0]     // Vsebina iz naslova v r0 -> r1
42
43         ldr r0, =STEV2   // Naslov od STEV1 -> r0
44         ldr r2, [r0]     // Vsebina iz naslova v r0 -> r2
45
46         add r3,r1,r2     // r1 + r2 -> r3
47
48         ldr r0, =VSOTA   // Naslov od STEV1 -> r0
49         str r3,[r0]      // iz registra r3 -> na naslov v r0
50
51 __end:  b    __end
52
```

Delo na STM32F4 razvojnem sistemu

Priključitev :

- **Mini USB** priklop na **krajši stranici**, svetita rdeči **LED** diodi

Poseben začetni projekt za *STM32F4* (e-učilnica) :

- ***dodajanje vsebine (template.s):***

'template.s - STM32CubeIDE

avigate Search Project Run Window Help

template.s

```
54
55 _start:
56     // Enable GPIOD Peripheral Clock (bit 3 in AHB1ENR register)
57     ldr r6, = RCC_AHB1ENR       // Load peripheral clock reg address to r6
58     ldr r5, [r6]                // Read its content to r5
59     orr r5, #0x00000008          // Set bit 3 to enable GPIOD clock
60     str r5, [r6]                // Store result in peripheral clock register
61
62     // Make GPIOD Pin12 as output pin (bits 25:24 in MODER register)
63     ldr r6, = GPIOD_MODER       // Load GPIOD MODER register address to r6
64     ldr r5, [r6]                // Read its content to r5
65     bic r5, #0x3000000          // Clear bits 24, 25 for P12
66     orr r5, #0x01000000         // Write 01 to bits 24, 25 for P12
67     str r5, [r6]                // Store result in GPIOD MODER register
68
69     // Set GPIOD Pin12 to 1 (bit 12 in ODR register)
70     ldr r6, = GPIOD_ODR         // Load GPIOD output data register
71     ldr r5, [r6]                // Read its content to r5
72     orr r5, #0x1000              // write 1 to pin 12
73     str r5, [r6]                // Store result in GPIOD output data register
74
75     // Set GPIOD Pin12 to 0 (bit 12 in ODR register)
76     ldr r6, = GPIOD_ODR         // Load GPIOD output data register
77     ldr r5, [r6]                // Read its content to r5
78     bic r5, #0x1000             // write 0 to pin 12
79     str r5, [r6]                // Store result in GPIOD output data register
80
81 loop:
82     nop                         // No operation. Do nothing.
83     b loop                      // Jump to loop
84
```

STM32 CubeIDE, STM32F4 (izbrana dokumentacij

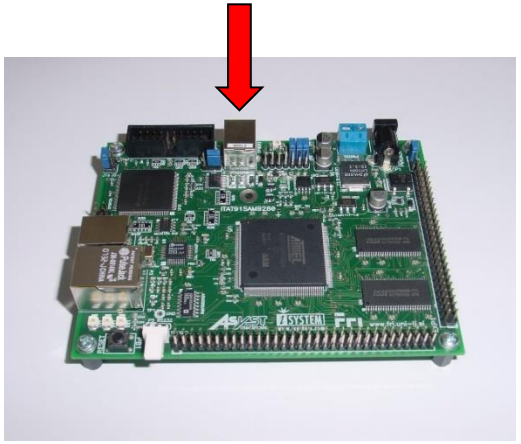
--------- Razvojni sistem ----------

- STM32 CubeIDE
- ORLab-STM32 - GitHub repozitorij
- User Manual Discovery kit stm32f407vg Uploaded 8/11/21, 12.58
- DataSheet_stm32f407vg Uploaded 8/11/21, 12.56
- Reference Manual rm0090-stm32f407417 Uploaded 8/11/21, 12.57
- Programming_Manual_pm0214-stm32-cortexm4-mcus-and-mpu
- Arm Cortex-M4 Processor Datasheet Short Uploaded 29/10/21, 15.00

--------- Cortex-M arhitektura, zbirnik ----------

- ARM Cortex-M for Beginners ARM 2017 Uploaded 29/10/21, 14.50

Delo na FRI-SMS razvojnem sistemu



Priključitev :

- **USB** priklop na **daljši stranici**, sveti **zelena LED** dioda

Poseben projekt za FRI-SMS (e-učilnica) :

- ***dodatne nastavitve*** (informativno) :
  - frekvenca urinega signala (višja poveča porabo!)
  - vklop predpomnilnikov
  - inicializacija sklada oz. SP – kazalca na sklad
- ***dodajanje vsebine (start.s):***
  - *podatki/operandi:*
    - dodamo v `/*constants*/` ,končamo z .align
  - *program* :
    - dodamo v `/* enter your code here */`
    - na koncu programa je mrtva zanka
    - podprograme dodamo za mrtvo zanko