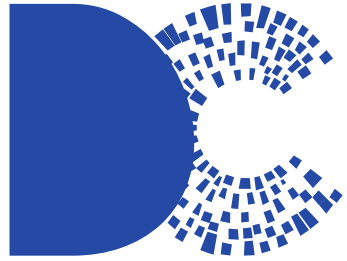
UNIVERSIDADE FEDERAL DE SERGIPE



DEPARTAMENTO DE COMPUTAÇÃO

Arquitetura ARM

Sistemas Embarcados

Bruno Prado

Departamento de Computação / UFS

Introdução

- Visão geral
  - Concebida em 1985, sob o nome do **A**corn **R**ISC **M**achine e posteriormente **A**dvanced **R**ISC **M**achine
  - Arquitetura RISC Harvard de 32 bits
  - Família de processadores Cortex-**A**, **R** e **M** com instruções de 16 (*Thumb*), 32 ou 64 bits

Introdução

- Visão geral
  - Concebida em 1985, sob o nome do **A**corn **R**ISC **M**achine e posteriormente **A**dvanced **R**ISC **M**achine
  - Arquitetura RISC Harvard de 32 bits
  - Família de processadores Cortex-**A**, **R** e **M** com instruções de 16 (*Thumb*), 32 ou 64 bits
    - **A**: alto desempenho e propósito geral

Introdução

- Visão geral
  - Concebida em 1985, sob o nome do **A**corn **R**ISC **M**achine e posteriormente **A**dvanced **R**ISC **M**achine
  - Arquitetura RISC Harvard de 32 bits
  - Família de processadores Cortex-**A**, **R** e **M** com instruções de 16 (*Thumb*), 32 ou 64 bits
    - **A**: alto desempenho e propósito geral
    - **R**: segurança crítica e de tempo real rígido

Introdução

- Visão geral
  - Concebida em 1985, sob o nome do **A**corn **R**ISC **M**achine e posteriormente **A**dvanced **R**ISC **M**achine
  - Arquitetura RISC Harvard de 32 bits
  - Família de processadores Cortex-**A**, **R** e **M** com instruções de 16 (*Thumb*), 32 ou 64 bits
    - **A**: alto desempenho e propósito geral
    - **R**: segurança crítica e de tempo real rígido
    - **M**: baixo custo e sistemas embarcados

Introdução

- Visão geral
  - Concebida em 1985, sob o nome do **A**corn **R**ISC **M**achine e posteriormente **A**dvanced **R**ISC **M**achine
  - Arquitetura RISC Harvard de 32 bits
  - Família de processadores Cortex-**A**, **R** e **M** com instruções de 16 (*Thumb*), 32 ou 64 bits
    - **A**: alto desempenho e propósito geral
    - **R**: segurança crítica e de tempo real rígido
    - **M**: baixo custo e sistemas embarcados
  - Modelo de negócios *fabless* para licença de componentes de *Intelectual Property* (IP)

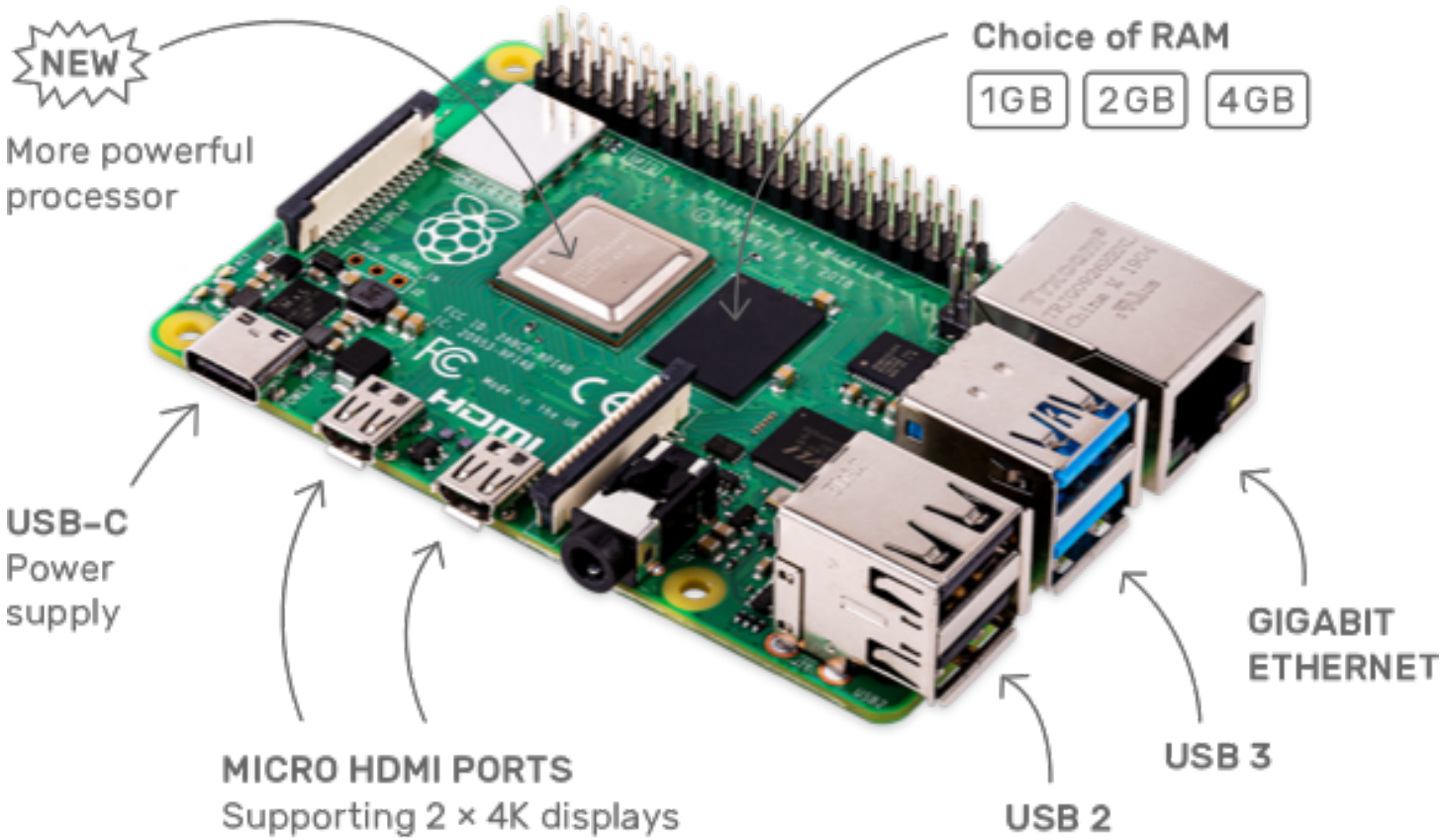
Introdução

- Visão geral
  - Concebida em 1985, sob o nome do **A**corn **R**ISC **M**achine e posteriormente **A**dvanced **R**ISC **M**achine
  - Arquitetura RISC Harvard de 32 bits
  - Família de processadores Cortex-**A**, **R** e **M** com instruções de 16 (*Thumb*), 32 ou 64 bits
    - **A**: alto desempenho e propósito geral
    - **R**: segurança crítica e de tempo real rígido
    - **M**: baixo custo e sistemas embarcados
  - Modelo de negócios *fabless* para licença de componentes de *Intelectual Property* (IP)

Mais de 150 bilhões de dispositivos produzidos até 2019

Introdução

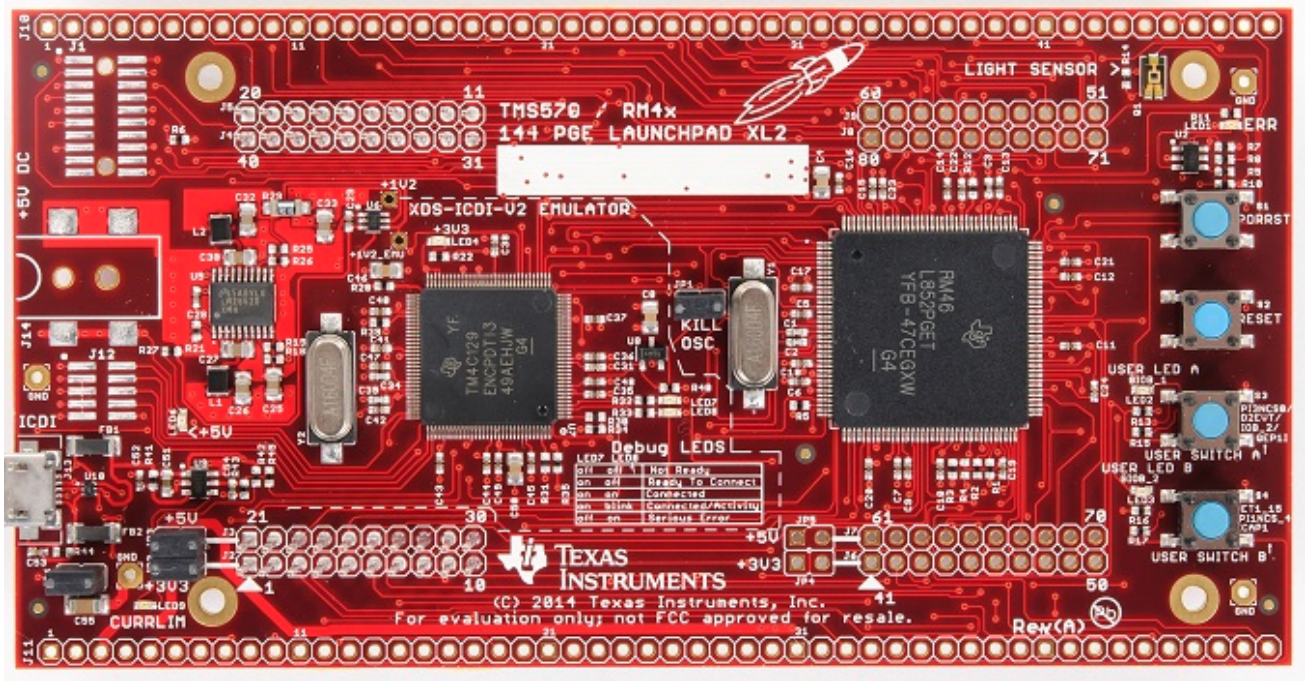
- Raspberry Pi 4
  - ARM Cortex-A72 (ARMv8) de 64 bits @ 1.5 GHz
  - Custo > US$ 35
  - Temperatura de operação entre 0° C e 50° C
  - Voltagem: 5 V (corrente mínima de 3 A)



Fonte: https://www.raspberrypi.org/products/raspberry-pi-4-model-b

Introdução

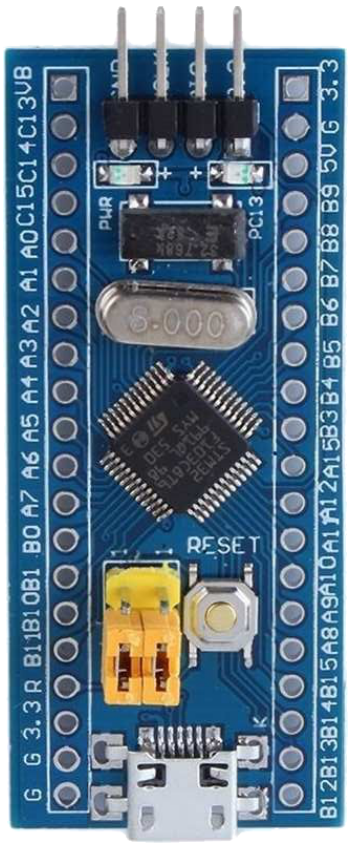
- Placa de desenvolvimento TMS570
  - ARM Cortex-R4 (ARMv7-R) de 32 bits @ 220 MHz
  - Custo > US$ 50
  - Temperatura de operação entre -40° C e +105° C
  - Voltagem: 3,0 V até 3,6 V (225 mA até 485 mA)

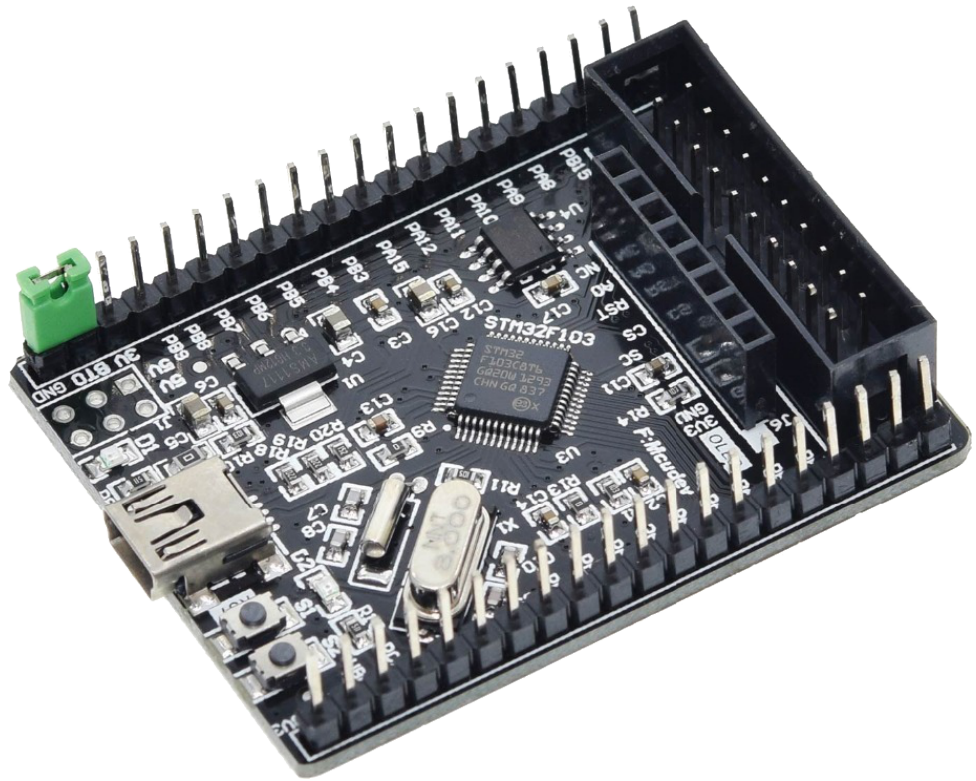


Fonte: http://www.ti.com/tool/LAUNCHXL2-RM46

Introdução

- Placa de desenvolvimento STM32F103C8
  - ARM Cortex-M3 (ARMv7-M) de 32 bits @ 72 MHz
  - Custo < US$ 4
  - Temperatura de operação entre -40° C e +125° C
  - Voltagem: 2,0 V até 3,6 V (5,5 mA até 50,3 mA)

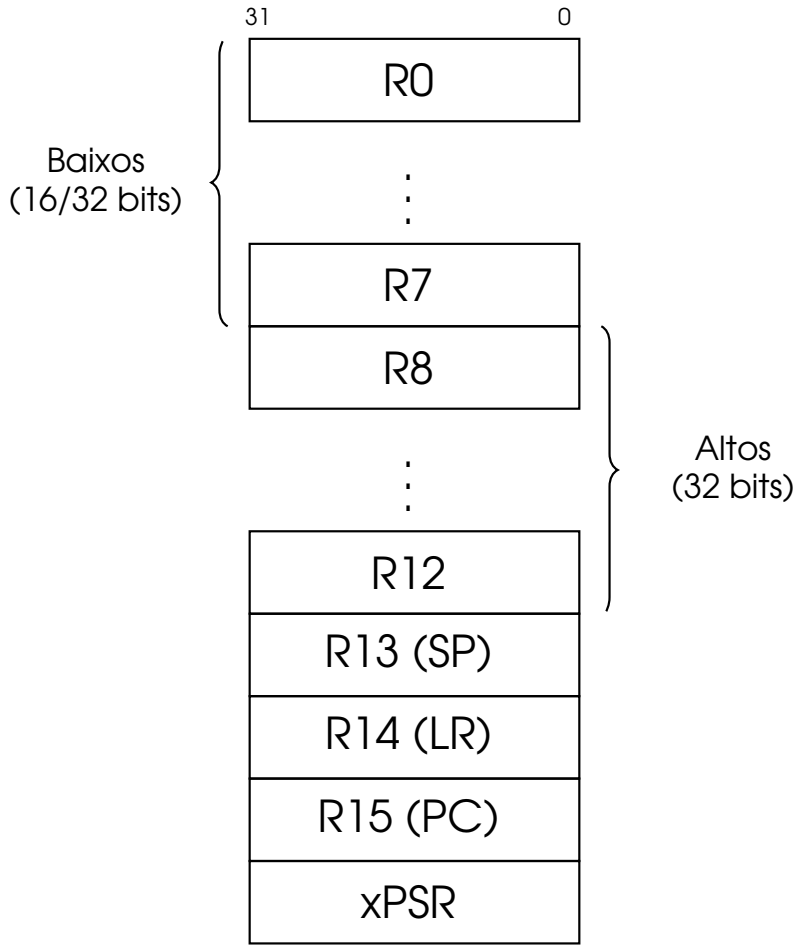




Fonte: https://www.st.com/resource/en/datasheet/stm32f103c8.pdf

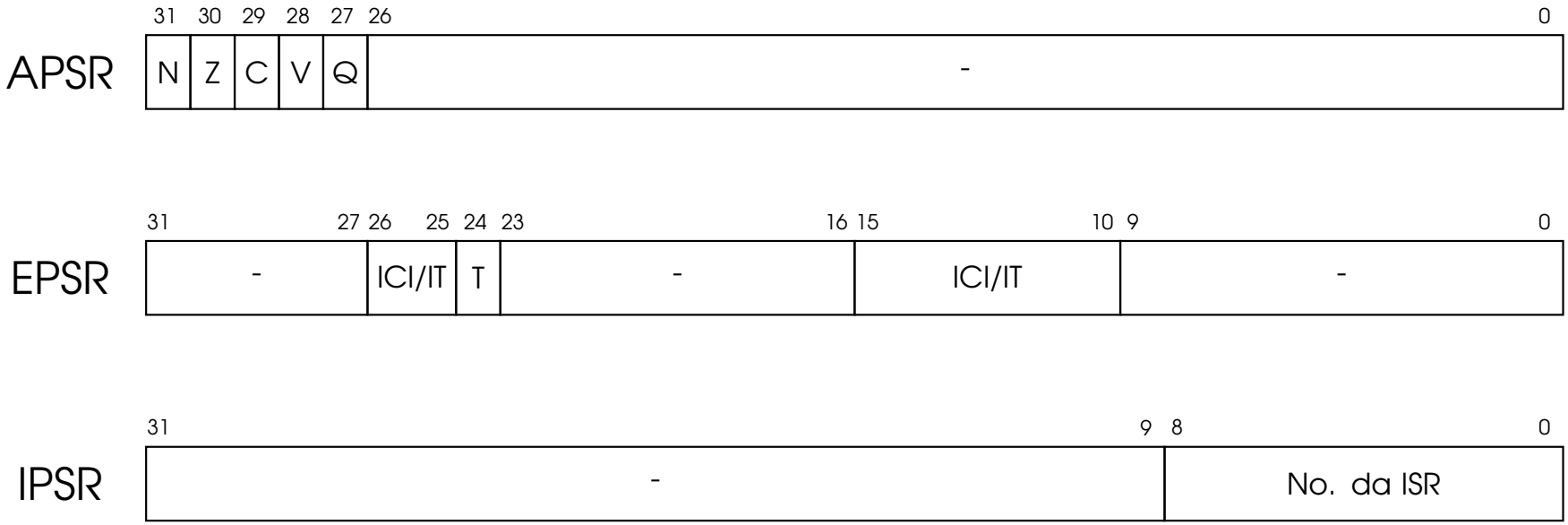
Registradores

- 13 registradores de propósito geral (R0 - R12), ponteiro da pilha (R13 ou SP), endereço de retorno ou *link* (R14 ou LR) e contador de programa (R15 ou PC)



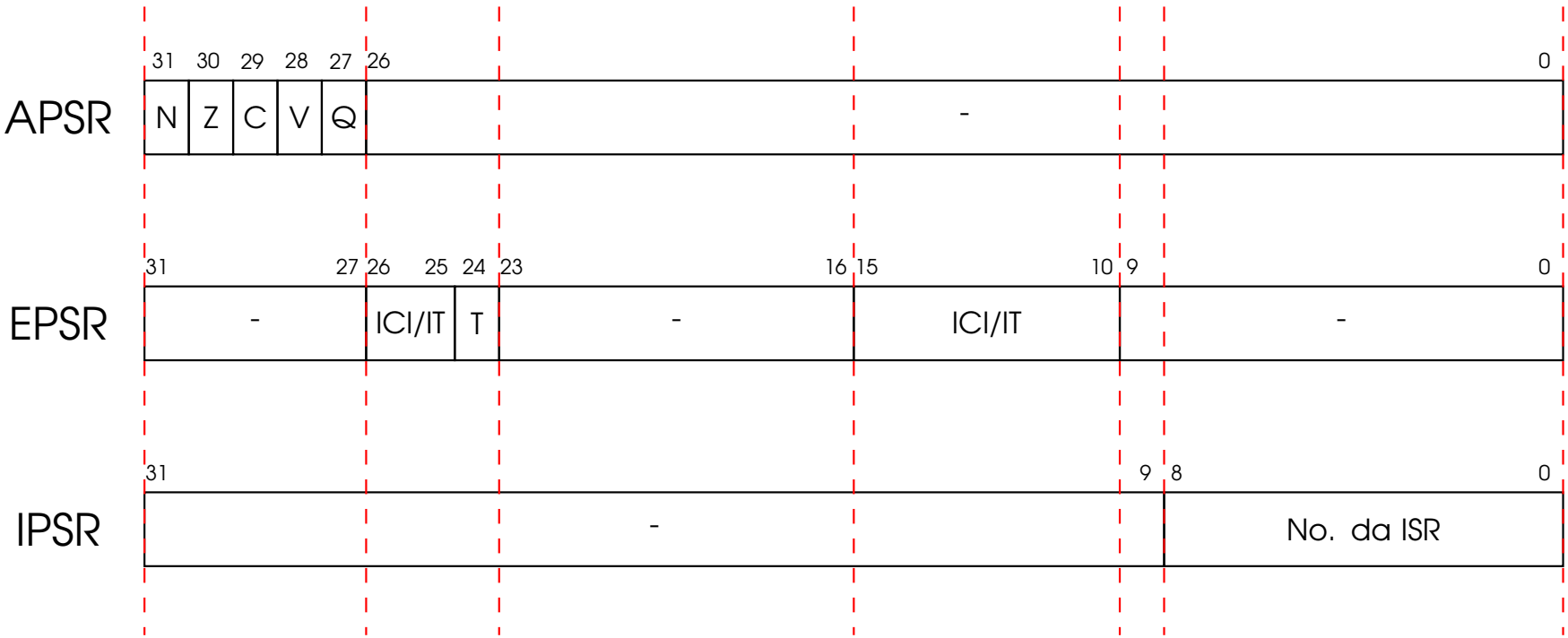
Registradores

- Status do programa para condições da aplicação (APSR), controle de execução (EPSR) e para número de rotina de tratamento de interrupção (IPSR)



Registradores

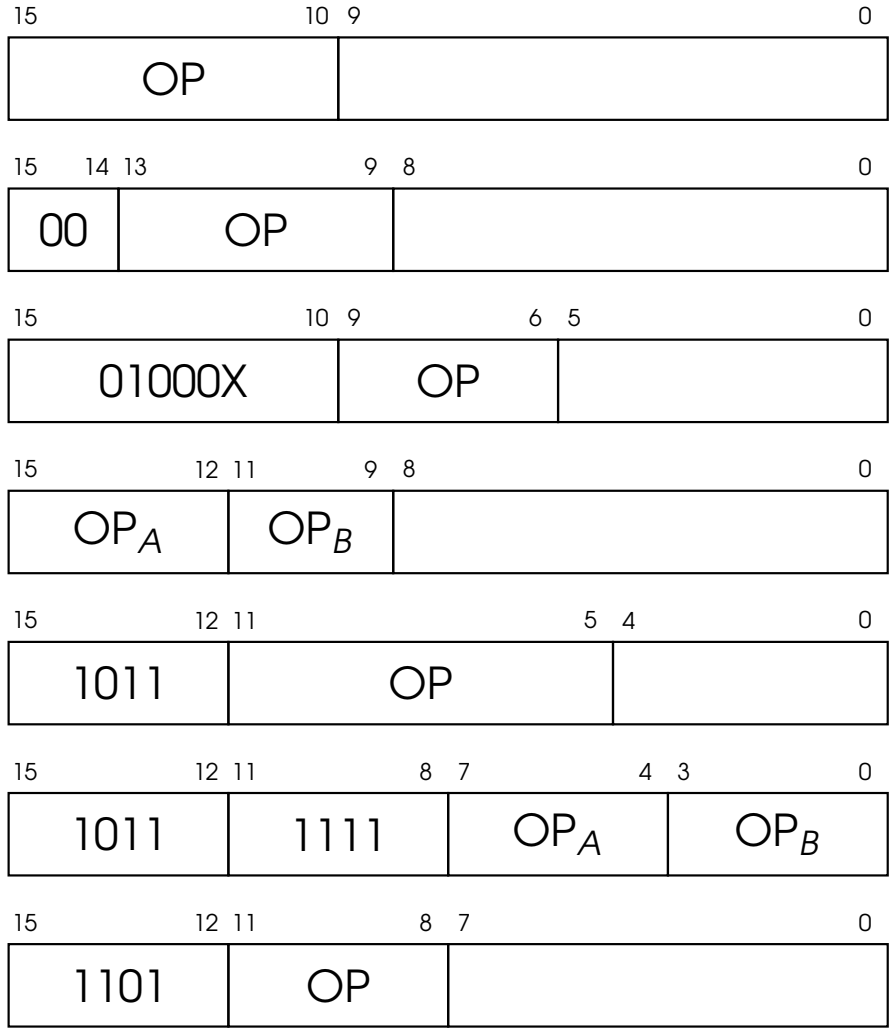
- Status do programa para condições da aplicação (APSR), controle de execução (EPSR) e para número de rotina de tratamento de interrupção (IPSR)



Os campos não estão sobrepostos
e podem ser acessados individualmente

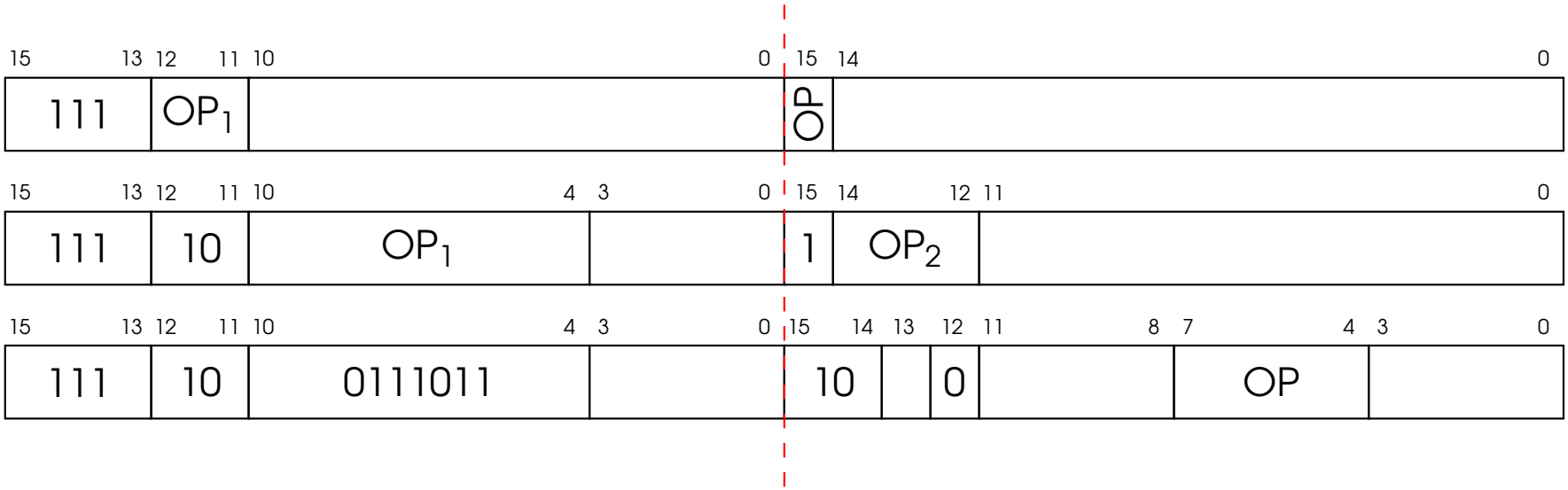
Modo Thumb

- Redução do tamanho do programa (perfil M)
  - Instruções com alinhamento de 16 bits



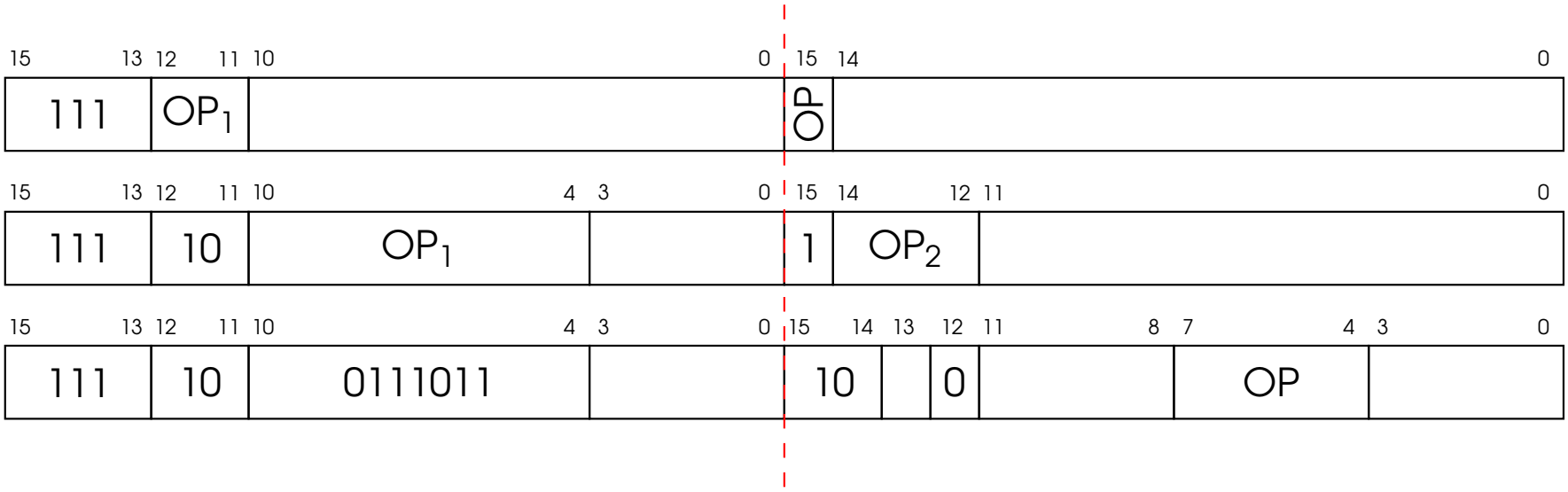
Modo Thumb

- Redução do tamanho do programa (perfil M)
  - Instruções com alinhamento de 16 bits



Modo Thumb

- Redução do tamanho do programa (perfil M)
  - Instruções com alinhamento de 16 bits



65% do tamanho do código de 32 bits
e 160% mais rápido em uma memória de 16 bits

Modos de inicialização

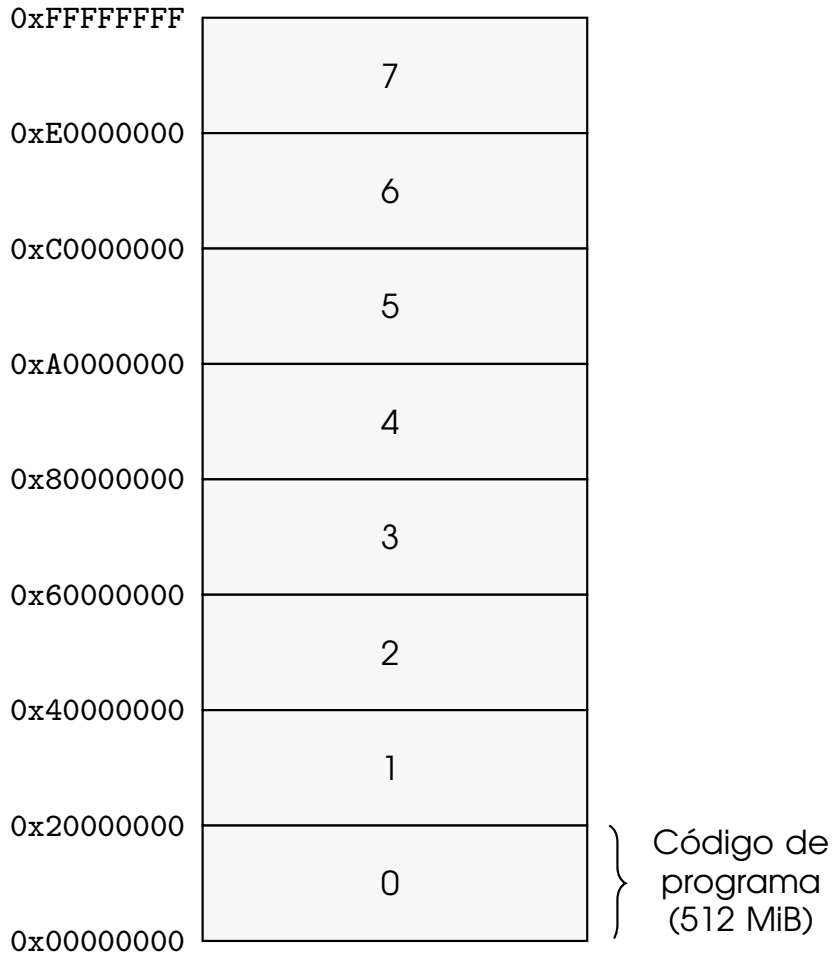
- Diferentes modos de inicialização (*boot*) podem ser utilizados pelo *bootloader* armazenado na memória de sistema (ROM) para programar a memória FLASH
  - Programação via USART1 ($RX = PA10$ e $TX = PA9$)

Memória	*BOOT1*	*BOOT0*	Endereço	Mapeamento
FLASH	*X*	0	0*x*08000000	0*x*00000000
Sistema	0	1	0*x*1*FFFF*000	0*x*00000000
SRAM	1	1	0*x*20000000	–

Configurações de inicialização

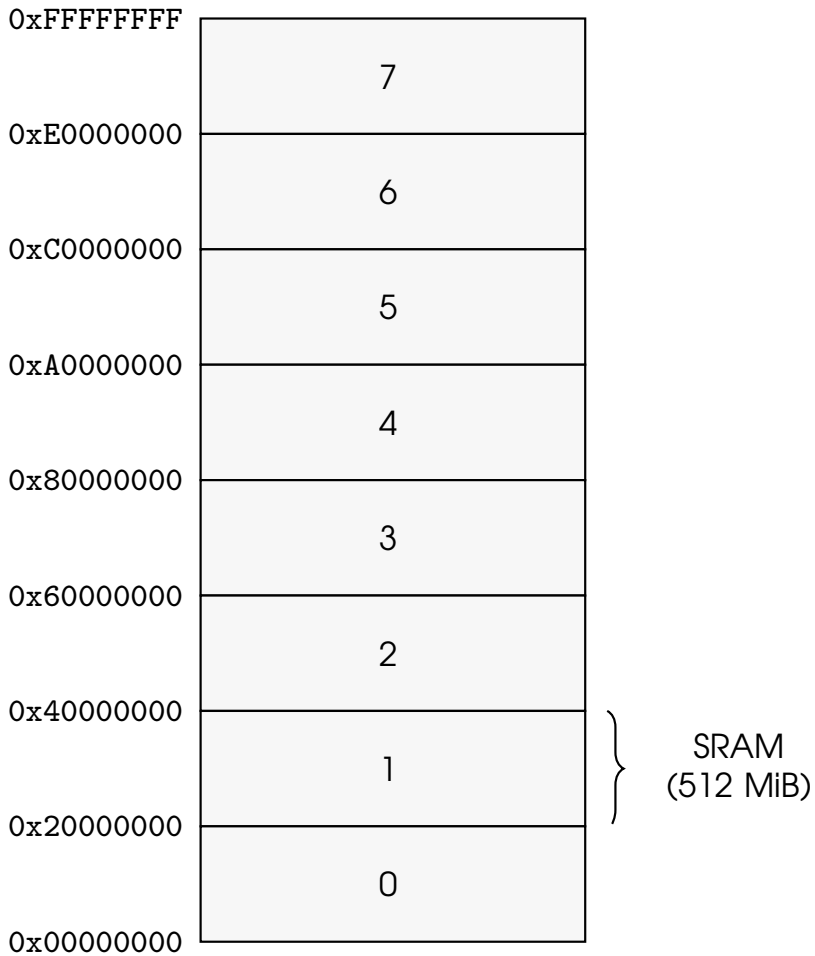
Mapa de memória

- Define como os dispositivos mapeados em memória e as informações estão organizados nos endereços



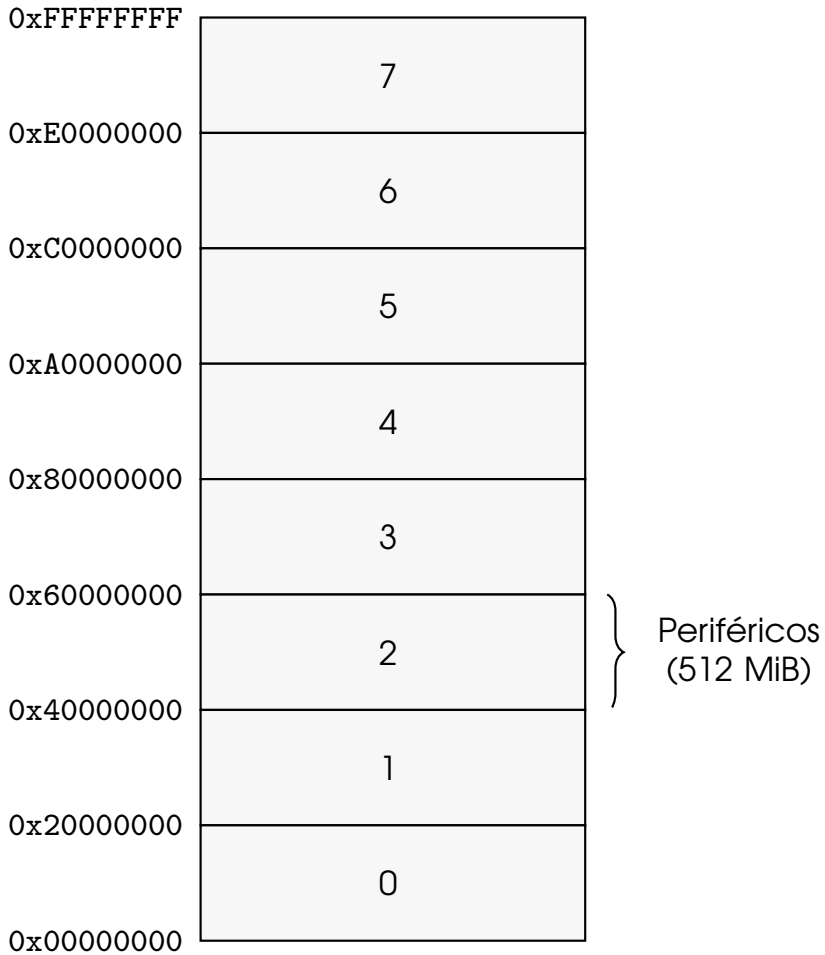
Mapa de memória

- Define como os dispositivos mapeados em memória e as informações estão organizados nos endereços



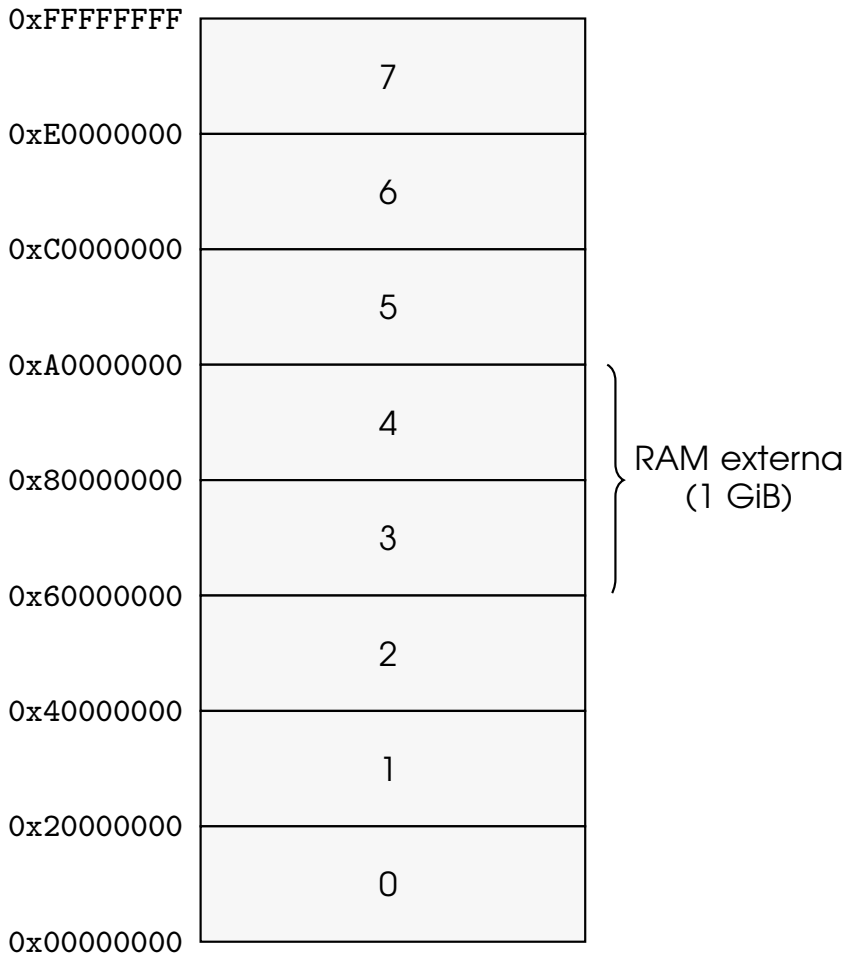
Mapa de memória

- Define como os dispositivos mapeados em memória e as informações estão organizados nos endereços



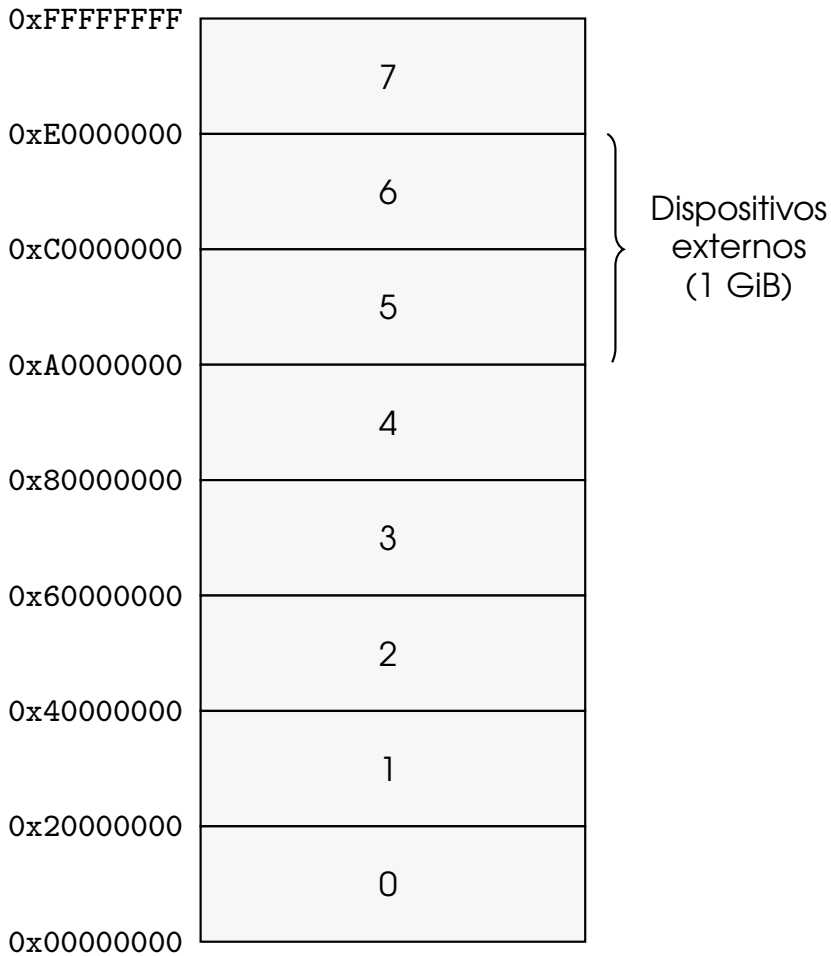
Mapa de memória

- Define como os dispositivos mapeados em memória e as informações estão organizados nos endereços



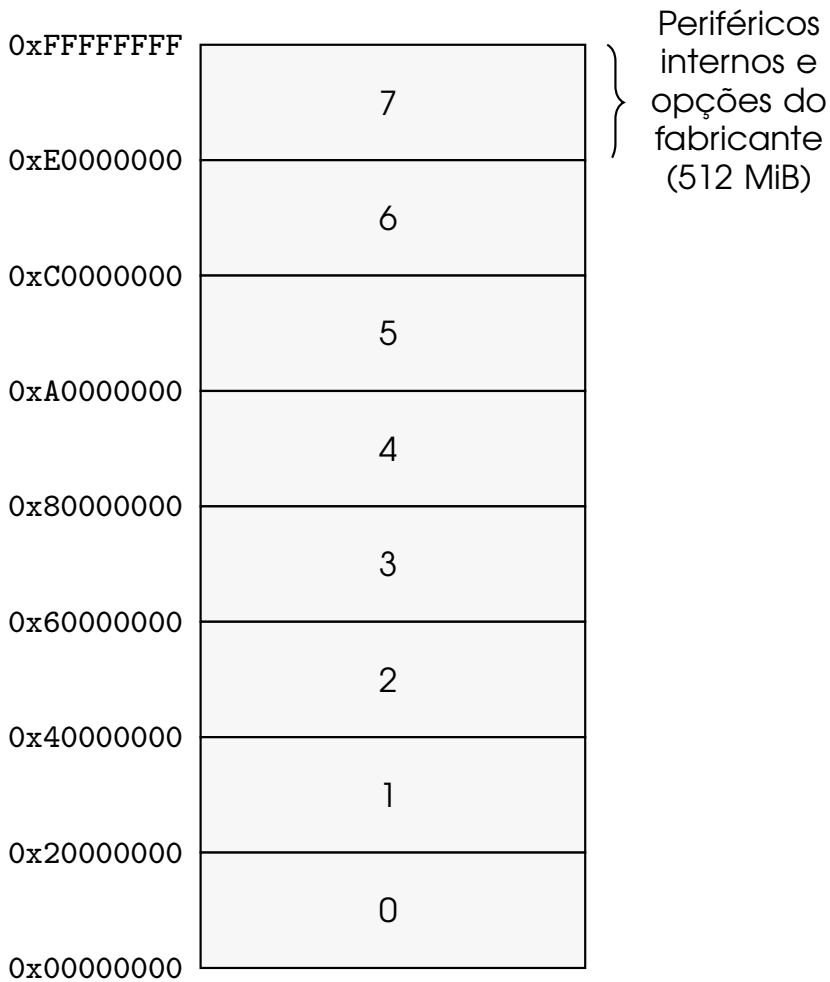
Mapa de memória

- Define como os dispositivos mapeados em memória e as informações estão organizados nos endereços



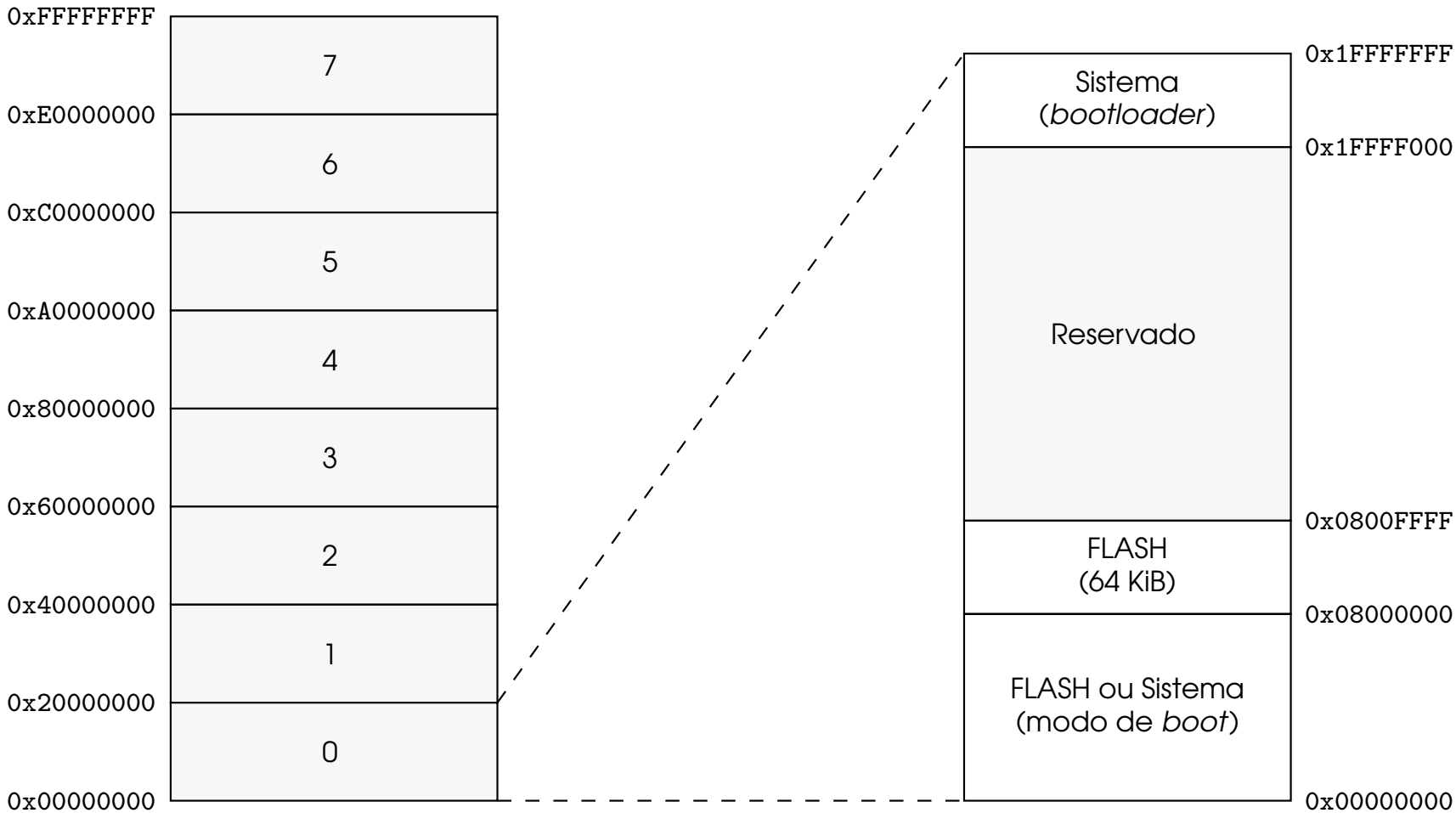
Mapa de memória

- Define como os dispositivos mapeados em memória e as informações estão organizados nos endereços



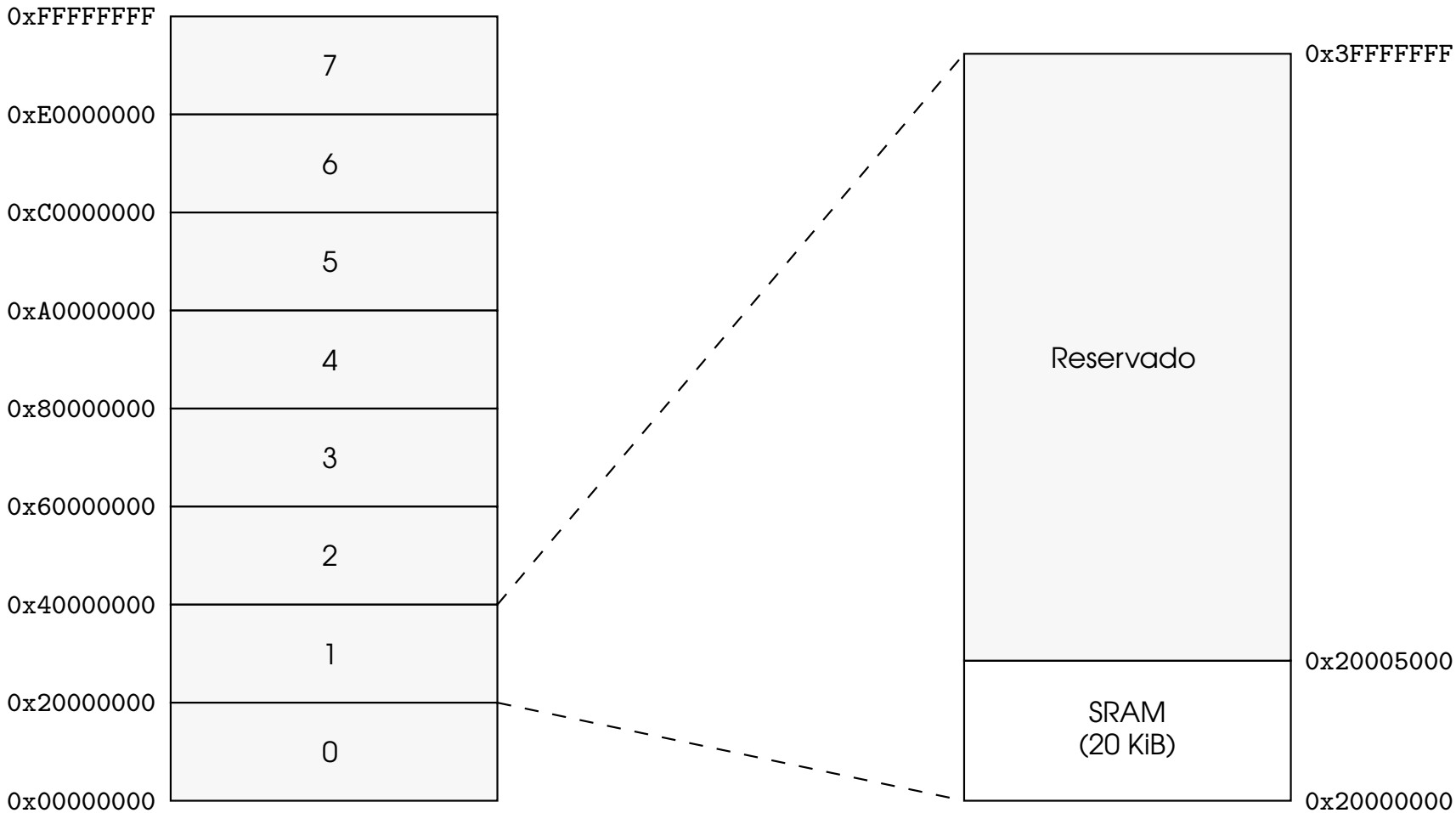
Mapa de memória

- Define como os dispositivos mapeados em memória e as informações estão organizados nos endereços



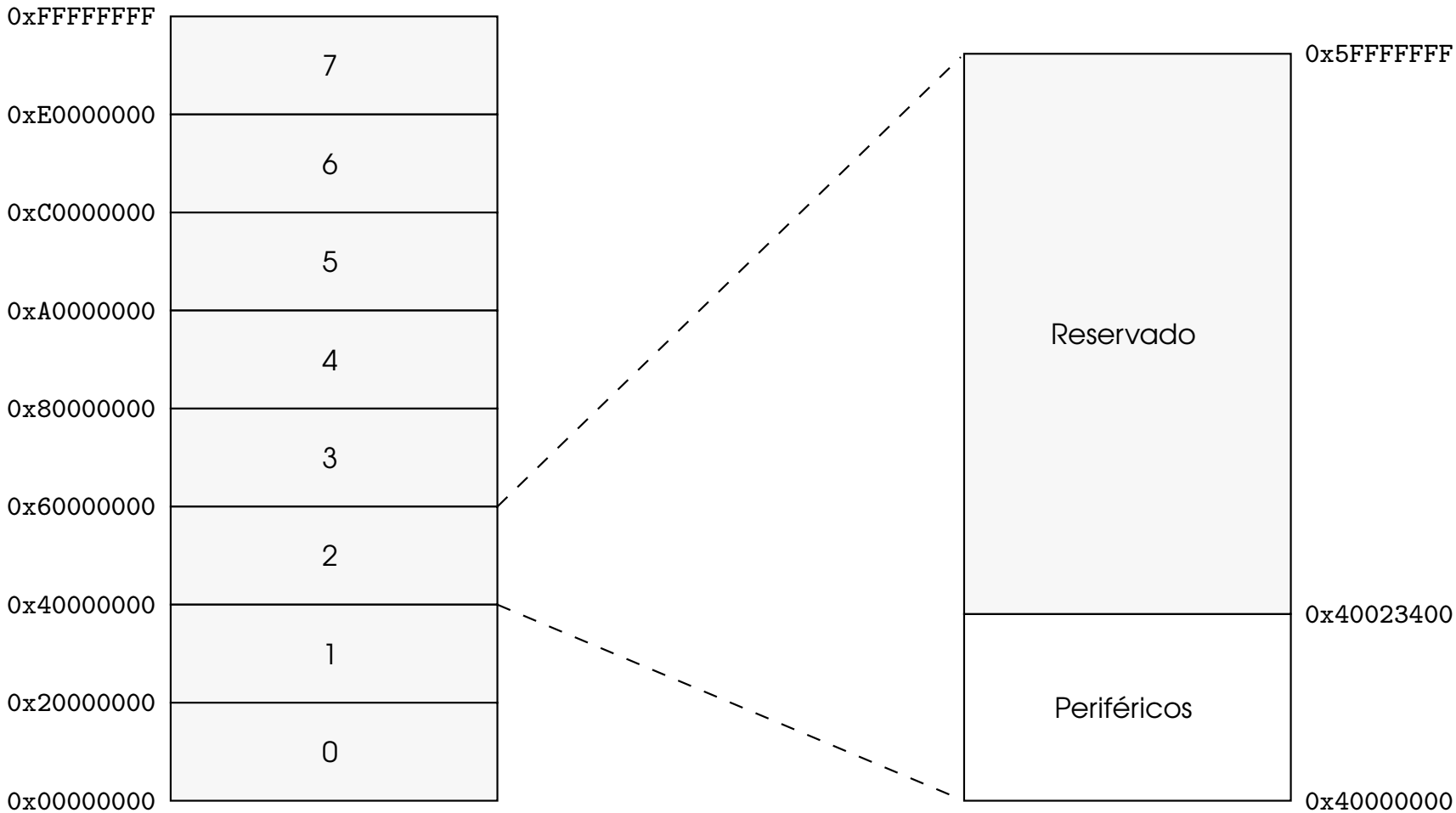
Mapa de memória

- Define como os dispositivos mapeados em memória e as informações estão organizados nos endereços



Mapa de memória

- Define como os dispositivos mapeados em memória e as informações estão organizados nos endereços



Mapa de memória

- Define como os dispositivos mapeados em memória e as informações estão organizados nos endereços

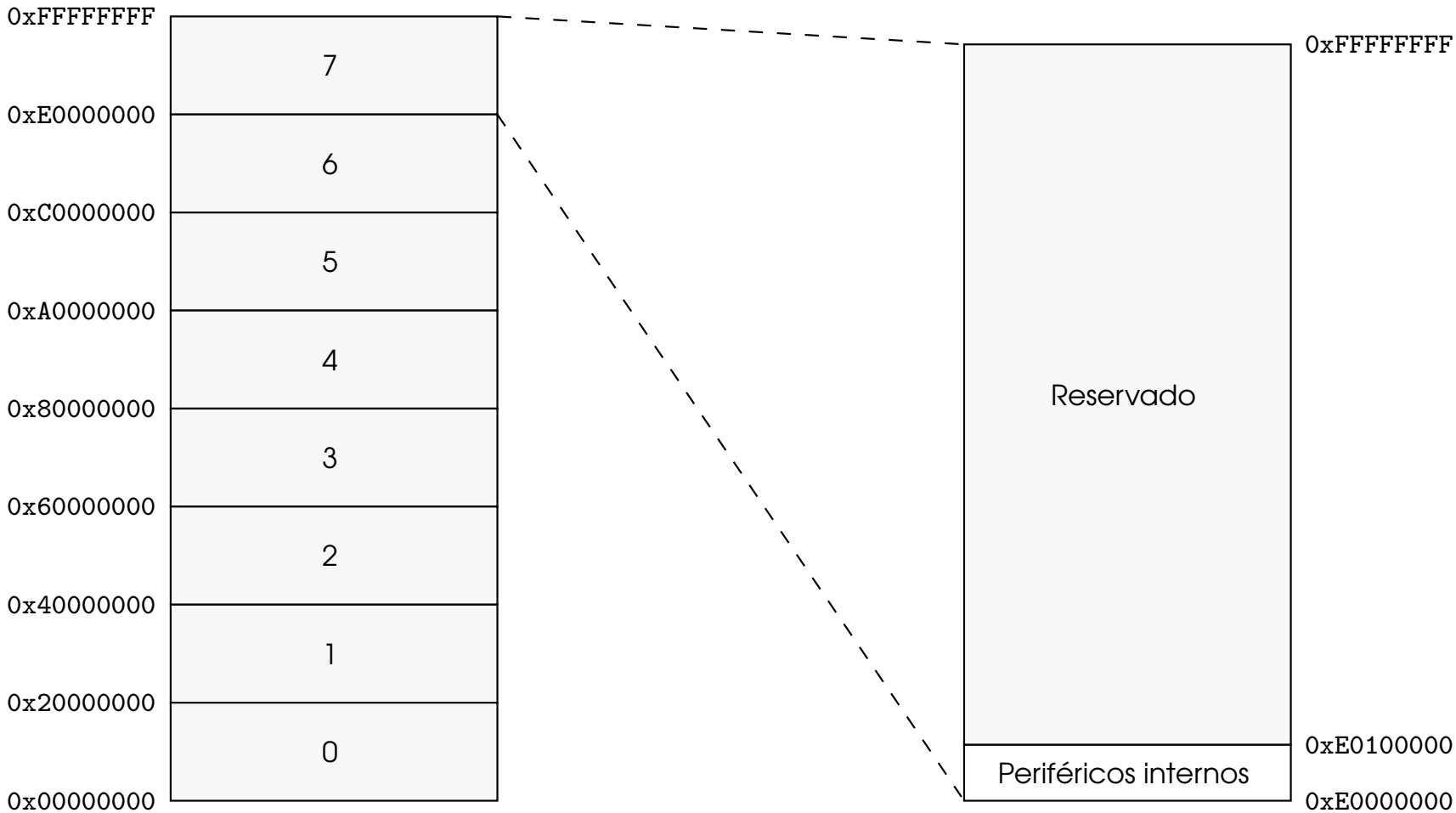


Tabela de vetor

- Inicialização do sistema (start.s)

```
1  .syntax unified
2  .cpu    cortex-m3
3  .fpu    softvfp
4  .thumb
5
6  .section .text.vector_table
7  .global  _vector_table
8  _vector_table:
9      // Topo da pilha
10     .word _estack
11     // Rotina de inicialização
12     .word _start
13
14 .section .text.start
15 .thumb_func
16 _start:
17     bl main
18     b  .
```

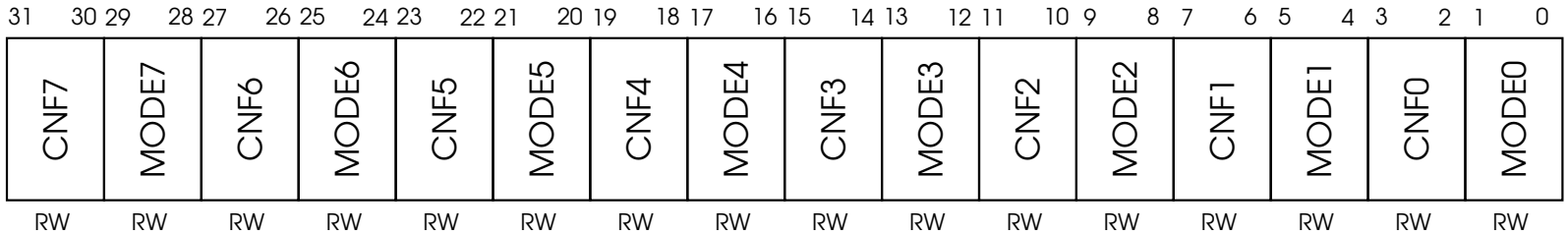
Script de linkagem

- Mapeamento das seções (stm32f103c8.ld)

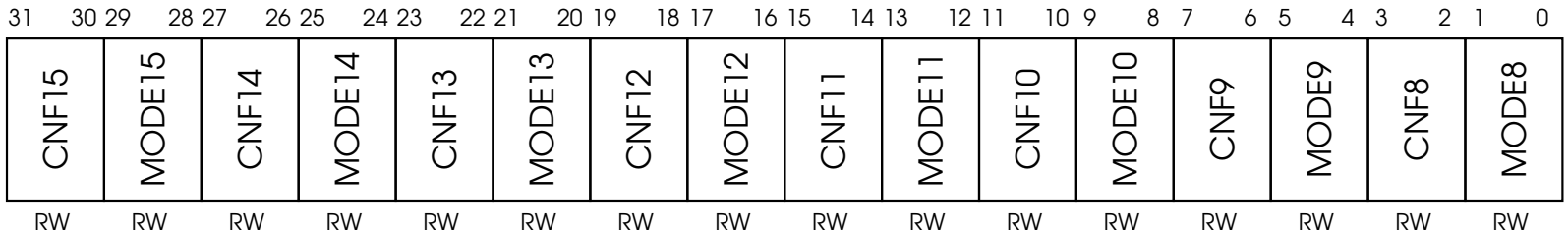
```
1  FLASH_init = 0x08000000;
2  FLASH_size = 64K;
3  RAM_init   = 0x20000000;
4  RAM_size   = 20K;
5  _estack    = RAM_init + RAM_size;
6
7  MEMORY {
8      FLASH : ORIGIN = FLASH_init, LENGTH = FLASH_size
9      RAM   : ORIGIN = RAM_init,   LENGTH = RAM_size
10 }
11
12 SECTIONS {
13     .text   : { *(.text.vector_table) *(.text.start)
           *(.text*) } > FLASH
14     .rodata : { *(.rodata*) } > FLASH
15     .data   : { *(.data*) } > RAM AT > FLASH
16     .bss    : { *(.bss*) } > RAM
17 }
```

Registradores de GPIO

- Modo de operação e direção dos pinos (y) de propósito geral (GPIO) das portas A até G (x)
  - Registrador de configuração GPIOx_CRL
    - *Offset* $= 0x00$
    - *Valor inicial* $= 0x44444444$



  - Registrador de configuração GPIOx_CRH
    - *Offset* $= 0x04$
    - *Valor inicial* $= 0x44444444$



Registradores de GPIO

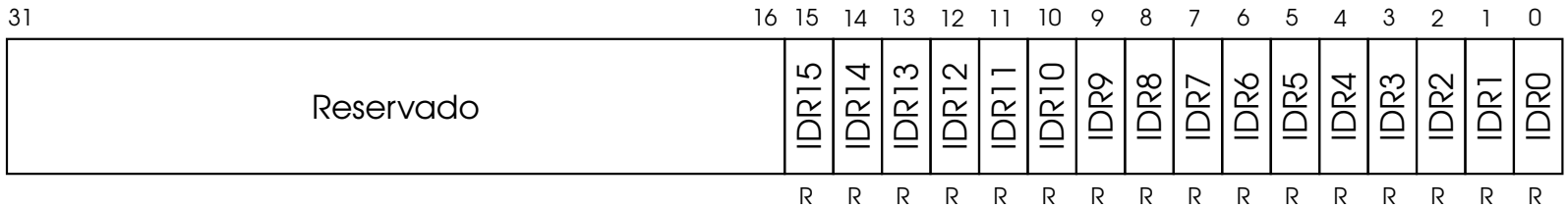
- Modo de operação e direção dos pinos (y) de propósito geral (GPIO) das portas A até G (x)
  - 16 pinos por porta

CNFy	MODEy = 00 (Entrada)	MODEy > 00 (Saída)
00	Analógico	Propósito geral (*push-pull*)
01	Flutuando	Propósito geral (*open-drain*)
10	Entrada (*pull-down/up*)	Função alternativa (*push-pull*)
11	Reservado	Função alternativa (*open-drain*)

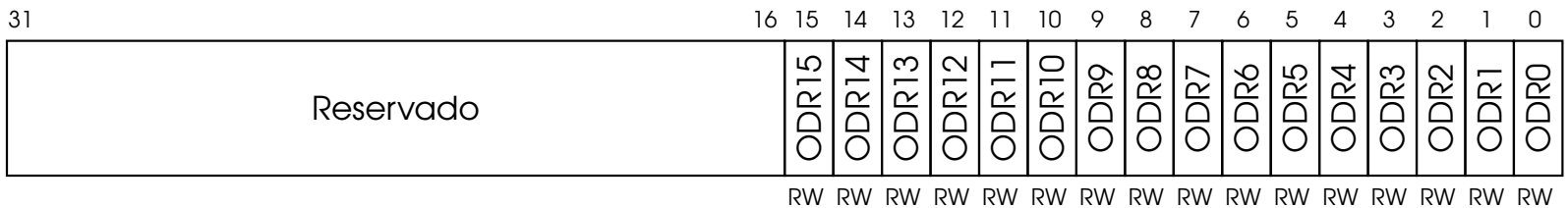
MODEy	Configuração
00	Entrada
01	Saída (10 MHz)
10	Saída (2 MHz)
11	Saída (50 MHz)

Registradores de GPIO

- Entrada e saída de dados dos pinos (y) de propósito geral (GPIO) das portas A até G (x)
  - Registrador de entrada de dados (GPIOx_IDR)
    - *Offset* $= 0x08$
    - *Valor inicial* $= 0x0000XXXX$

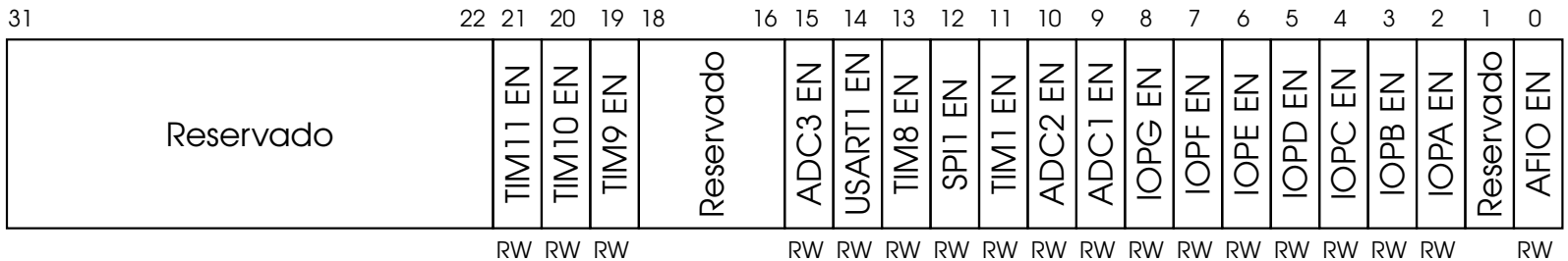


  - Registrador de saída de dados (GPIOx_ODR)
    - *Offset* $= 0x0C$
    - *Valor inicial* $= 0x00000000$



Registradores RCC

- Controle de reinicialização e de relógio
  - Registrador de habilitação APB2 (RCC_APB2ENR)
    - *Offset* = 0x18
    - *Valor inicial* = 0x00000000



Desenvolvimento

- Ligando/desligando LED (blink.c)
  - Macro para espera em milissegundos

```
1   // Tipos inteiros de tamanho fixo
2   #include <stdint.h>
3   // Macro de delay em ms (8 MHz)
4   #define delay_ms(X)\
5       asm volatile(\
6           "mov␣r1,␣%[counter]\n\t"
7           "loop:\n\t"\
8           "␣␣␣subs␣r1,␣r1,␣1\n\t"\
9           "␣␣␣bne␣loop"\
10          :\
11          : [counter] "r"(X * 4000)\
12          : "r1"\
13      )
...
...
```

Desenvolvimento

- Ligando/desligando LED (blink.c)
  - Definição dos campos e dos registradores

```
1   // Tipos inteiros de tamanho fixo
2   #include <stdint.h>
... ...
14  // Definição de habilitação de GPIO C
15  #define GPIOC_EN   (4)
16  // Definições do pino 13 de GPIO C
17  #define PC13_CNF   (22)
18  #define PC13_MODE  (20)
19  #define PC13_ODR   (13)
20  // Reset and Clock Control (RCC)
21  volatile uint32_t* const RCC_APB2ENR = (volatile
        uint32_t*)(0x40021018);
22  // Porta C
23  volatile uint32_t* const GPIOCCRH = (volatile
        uint32_t*)(0x40011004);
24  volatile uint32_t* const GPIOCODR = (volatile
        uint32_t*)(0x4001100C);
... ...
```

Desenvolvimento

- Ligando/desligando LED (blink.c)
  - Inicialização e controle do pino PC13 (LED)

```
 1  // Tipos inteiros de tamanho fixo
 2  #include <stdint.h>
...  ...
25  // Função principal
26  int main() {
27      // Habilitando clock do GPIO C
28      (*RCC_APB2ENR) |= 1 << GPIOC_EN;
29      // Ajustando PC13 como saída (2 MHz)
30      (*GPIOCCRH) &= ~(0b11 << PC13_CNF);
31      (*GPIOCCRH) &= ~(0b11 << PC13_MODE);
32      (*GPIOCCRH) |= (0b10 << PC13_MODE);
33      // Laço infinito
34      while(1) {
35          // Invertendo valor do pino PC13 por 1 segundo
36          (*GPIOCODR) ^= (1 << PC13_ODR);
37          delay_ms(1000);
38      }
39  }
```

Desenvolvimento

- Processo de compilação e programação

Desenvolvimento

- Processo de compilação e programação

```
$ arm-none-eabi-as -mcpu=cortex-m3 -mthumb -g start.s -o start.o
$
```

Desenvolvimento

- Processo de compilação e programação

```
$ arm-none-eabi-as -mcpu=cortex-m3 -mthumb -g start.s -o start.o
$ arm-none-eabi-gcc -Wall -Os -mcpu=cortex-m3 -mthumb -g -nostdlib -n
ostartfiles -c blink.c -o blink.o
$
```

Desenvolvimento

- Processo de compilação e programação

```
$ arm-none-eabi-as -mcpu=cortex-m3 -mthumb -g start.s -o start.o
$ arm-none-eabi-gcc -Wall -Os -mcpu=cortex-m3 -mthumb -g -nostdlib -n
ostartfiles -c blink.c -o blink.o
$ arm-none-eabi-gcc -Wall -Os -mcpu=cortex-m3 -mthumb -g -nostdlib
-nostartfiles -T stm32f103c8.ld start.o blink.o -o blink.elf
$
```

Desenvolvimento

- Processo de compilação e programação

```
$ arm-none-eabi-as -mcpu=cortex-m3 -mthumb -g start.s -o start.o
$ arm-none-eabi-gcc -Wall -Os -mcpu=cortex-m3 -mthumb -g -nostdlib -n
ostartfiles -c blink.c -o blink.o
$ arm-none-eabi-gcc -Wall -Os -mcpu=cortex-m3 -mthumb -g -nostdlib
-nostartfiles -T stm32f103c8.ld start.o blink.o -o blink.elf
$ arm-none-eabi-objcopy -O binary blink.elf blink.bin
$
```

Desenvolvimento

- Processo de compilação e programação

```
$ arm-none-eabi-as -mcpu=cortex-m3 -mthumb -g start.s -o start.o
$ arm-none-eabi-gcc -Wall -Os -mcpu=cortex-m3 -mthumb -g -nostdlib -n
ostartfiles -c blink.c -o blink.o
$ arm-none-eabi-gcc -Wall -Os -mcpu=cortex-m3 -mthumb -g -nostdlib
-nostartfiles -T stm32f103c8.ld start.o blink.o -o blink.elf
$ arm-none-eabi-objcopy -O binary blink.elf blink.bin
$ st-flash write blink.bin 0x08000000
```

Desenvolvimento

- Processo de compilação e programação

```
$ st-flash write blink.bin 0x08000000
st-flash 1.5.1
2019-12-01T17:28:52 INFO usb.c: -- exit_dfu_mode
2019-12-01T17:28:52 INFO common.c: Loading device parameters....
2019-12-01T17:28:52 INFO common.c: Device connected is: F1 Medium-den
sity device, id 0x20036410
2019-12-01T17:28:52 INFO common.c: SRAM size: 0x5000 bytes (20 KiB),
Flash: 0x10000 bytes (64 KiB) in pages of 1024 bytes
2019-12-01T17:28:52 INFO common.c: Attempting to write 96 (0x60) byte
s to stm32 address: 134217728 (0x8000000)
Flash page at addr: 0x08000000 erased
2019-12-01T17:28:52 INFO common.c: Finished erasing 1 pages of 1024 (
0x400) bytes
2019-12-01T17:28:52 INFO common.c: Starting Flash write for VL/F0/F3/
F1_XL core id
2019-12-01T17:28:52 INFO flash_loader.c: Successfully loaded flash lo
ader in sram
  1/1 pages written
2019-12-01T17:28:52 INFO common.c: Starting verification of write com
```

Desenvolvimento

- Processo de compilação e programação

```
2019-12-01T17:28:52 INFO common.c: Device connected is: F1 Medium-den
sity device, id 0x20036410
2019-12-01T17:28:52 INFO common.c: SRAM size: 0x5000 bytes (20 KiB),
Flash: 0x10000 bytes (64 KiB) in pages of 1024 bytes
2019-12-01T17:28:52 INFO common.c: Attempting to write 96 (0x60
) bytes to stm32 address: 134217728 (0x8000000)
Flash page at addr: 0x08000000 erased
2019-12-01T17:28:52 INFO common.c: Finished erasing 1 pages of 1024 (
0x400) bytes
2019-12-01T17:28:52 INFO common.c: Starting Flash write for VL/F0/F3/
F1_XL core id
2019-12-01T17:28:52 INFO flash_loader.c: Successfully loaded flash lo
ader in sram
  1/1 pages written
2019-12-01T17:28:52 INFO common.c: Starting verification of write com
plete
2019-12-01T17:28:52 INFO common.c: Flash written and verified! jolly
good!
$
```

Desenvolvimento

- Processo de compilação e programação

```
2019-12-01T17:28:52 INFO common.c: Device connected is: F1 Medium-den
sity device, id 0x20036410
2019-12-01T17:28:52 INFO common.c: SRAM size: 0x5000 bytes (20 KiB),
Flash: 0x10000 bytes (64 KiB) in pages of 1024 bytes
2019-12-01T17:28:52 INFO common.c: Attempting to write 96 (0x60
) bytes to stm32 address: 134217728 (0x8000000)
Flash page at addr: 0x08000000 erased
2019-12-01T17:28:52 INFO common.c: Finished erasing 1 pages of 1024 (
0x400) bytes
2019-12-01T17:28:52 INFO common.c: Starting Flash write for VL/F0/F3/
F1_XL core id
2019-12-01T17:28:52 INFO flash_loader.c: Successfully loaded flash lo
ader in sram
  1/1 pages written
2019-12-01T17:28:52 INFO common.c: Starting verification of write com
plete
2019-12-01T17:28:52 INFO common.c: Flash written and verified! jolly
good!
$ exit
```

Depuração

- Processo de depuração na placa (ST-LINK)

Depuração

- Processo de depuração na placa (ST-LINK)

```
$ openocd -s /usr/share/openocd/scripts/ -f interface/stlink-v2.cfg -
f target/stm32f1x.cfg
```

Depuração

- Processo de depuração na placa (ST-LINK)

```
$ openocd -s /usr/share/openocd/scripts/ -f interface/stlink-v2.cfg -
f target/stm32f1x.cfg
Open On-Chip Debugger 0.10.0
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
Info : auto-selecting first available session transport "hla_
swd". To override use 'transport select <transport>'.
Info : The selected transport took over low-level target control. The
results might differ compared to plain JTAG/SWD
adapter speed: 1000 kHz
...
Info : STLINK v2 JTAG v31 API v2 SWIM v7 VID 0x0483 PID 0x3748
Info : using stlink api v2
Info : Target voltage: 3.121076
Info : stm32f1x.cpu: hardware has 6 breakpoints, 4 watchpoints
Info : accepting 'gdb' connection on tcp/3333
Info : device id = 0x20036410 Info : flash size = 64kbytes
...
```

Depuração

- Processo de depuração na placa (ST-LINK)

```
$ arm-none-eabi-gdb blink.elf
```

Depuração

- Processo de depuração na placa (ST-LINK)

```
GNU gdb (GNU Tools for Arm Embedded Processors 8-2018-q4-major)
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/g
pl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=x86_64-apple-darwin10 --target=arm
-none-eabi".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from blink.elf...
(gdb)
```

Depuração

- Processo de depuração na placa (ST-LINK)

```
GNU gdb (GNU Tools for Arm Embedded Processors 8-2018-q4-major)
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/g
pl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=x86_64-apple-darwin10 --target=arm
-none-eabi".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from blink.elf...
(gdb) target extended-remote:3333
```

Depuração

- Processo de depuração na placa (ST-LINK)

```
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=x86_64-apple-darwin10 --target=arm
-none-eabi".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from blink.elf...
(gdb) target extended-remote:3333
Remote debugging using :3333
0x0800003e in main () at blink_arm.c:38
38                      delay_ms(1000);
(gdb)
```

Depuração

- Processo de depuração na placa (ST-LINK)

```
0x0800003e in main () at blink_arm.c:38
38                      delay_ms(1000);
(gdb) monitor reset halt
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x08000008 msp: 0x20005000
(gdb) bt
#0  0x0800003e in main () at blink_arm.c:38
(gdb) si
main () at blink.c:28
28 (*RCC_APB2ENR) |= 1 << GPIOC_EN;
(gdb)
30 (*GPIOCCRH) &= ~(0b11 << PC13_CNF);
(gdb)
31 (*GPIOCCRH) &= ~(0b11 << PC13_MODE);
(gdb)
32 (*GPIOCCRH) |= (0b10 << PC13_MODE);
(gdb)
36 (*GPIOCODR) ^= (1 << PC13_ODR);
(gdb)
```

Depuração

- Processo de depuração na placa (ST-LINK)

```
(gdb) monitor reset halt
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x08000008 msp: 0x20005000
(gdb) bt
#0  0x0800003e in main () at blink_arm.c:38
(gdb) si
main () at blink.c:28
28 (*RCC_APB2ENR) |= 1 << GPIOC_EN;
(gdb)
30 (*GPIOCCRH) &= ~(0b11 << PC13_CNF);
(gdb)
31 (*GPIOCCRH) &= ~(0b11 << PC13_MODE);
(gdb)
32 (*GPIOCCRH) |= (0b10 << PC13_MODE);
(gdb)
36 (*GPIOCODR) ^= (1 << PC13_ODR);
(gdb) p/x *GPIOCCRH
$1 = 0x44244444
(gdb)
```

Depuração

- Processo de depuração na placa (ST-LINK)

```
(gdb) monitor reset halt
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x08000008 msp: 0x20005000
(gdb) bt
#0  0x0800003e in main () at blink_arm.c:38
(gdb) si
main () at blink.c:28
28 (*RCC_APB2ENR) |= 1 << GPIOC_EN;
(gdb)
30 (*GPIOCCRH) &= ~(0b11 << PC13_CNF);
(gdb)
31 (*GPIOCCRH) &= ~(0b11 << PC13_MODE);
(gdb)
32 (*GPIOCCRH) |= (0b10 << PC13_MODE);
(gdb)
36 (*GPIOCODR) ^= (1 << PC13_ODR);
(gdb) p/x *GPIOCCRH
$1 = 0x44244444
(gdb) quit
```

Depuração

- Processo de depuração na placa (ST-LINK)

```
28 (*RCC_APB2ENR) |= 1 << GPIOC_EN;
(gdb)
30 (*GPIOCCRH) &= ~(0b11 << PC13_CNF);
(gdb)
31 (*GPIOCCRH) &= ~(0b11 << PC13_MODE);
(gdb)
32 (*GPIOCCRH) |= (0b10 << PC13_MODE);
(gdb)
36 (*GPIOCODR) ^= (1 << PC13_ODR);
(gdb) p/x *GPIOCCRH
$1 = 0x44244444
(gdb) quit
A debugging session is active.

Inferior 1 [Remote target] will be killed.

Quit anyway? (y or n) y
...
$
```

Depuração

- Processo de depuração na placa (ST-LINK)

```
28 (*RCC_APB2ENR) |= 1 << GPIOC_EN;
(gdb)
30 (*GPIOCCRH) &= ~(0b11 << PC13_CNF);
(gdb)
31 (*GPIOCCRH) &= ~(0b11 << PC13_MODE);
(gdb)
32 (*GPIOCCRH) |= (0b10 << PC13_MODE);
(gdb)
36 (*GPIOCODR) ^= (1 << PC13_ODR);
(gdb) p/x *GPIOCCRH
$1 = 0x44244444
(gdb) quit
A debugging session is active.

Inferior 1 [Remote target] will be killed.

Quit anyway? (y or n) y
...
$ exit
```

Exercício

- Estude e reproduza os experimentos vistos nesta aula
  - Analise os manuais técnicos (*datasheets*) dos microcontroladores STM32F10x
  - Verifique as configurações necessárias para mudar a frequência de operação de 8 MHz (oscilador interno) para 72 MHz (cristal externo de 8 MHz)
  - Utilizando o *framework* Arduino, implemente e execute um exemplo equivalente ao *blink*
  - Faça um comparativo de utilização de memória dos binários gerados utilizando a ferramenta arm-size