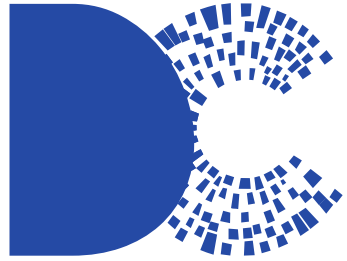
UNIVERSIDADE FEDERAL DE SERGIPE



DEPARTAMENTO DE COMPUTAÇÃO

Arquitetura Xtensa

Sistemas Embarcados

Bruno Prado

Departamento de Computação / UFS

Introdução

- Visão geral
  - Criada em 1999 pela Tensilica que depois foi comprada pela Cadence Design Systems em 2013
  - Arquitetura RISC de 32 bits extensível e configurável

Introdução

- Visão geral
  - Criada em 1999 pela Tensilica que depois foi comprada pela Cadence Design Systems em 2013
  - Arquitetura RISC de 32 bits extensível e configurável
    - Disponibilizado como código RTL sintetizável (IP)

Introdução

- Visão geral
  - Criada em 1999 pela Tensilica que depois foi comprada pela Cadence Design Systems em 2013
  - Arquitetura RISC de 32 bits extensível e configurável
    - Disponibilizado como código RTL sintetizável (IP)
    - Possibilita que funções ou instruções personalizadas sejam utilizadas de forma nativa no desenvolvimento

Introdução

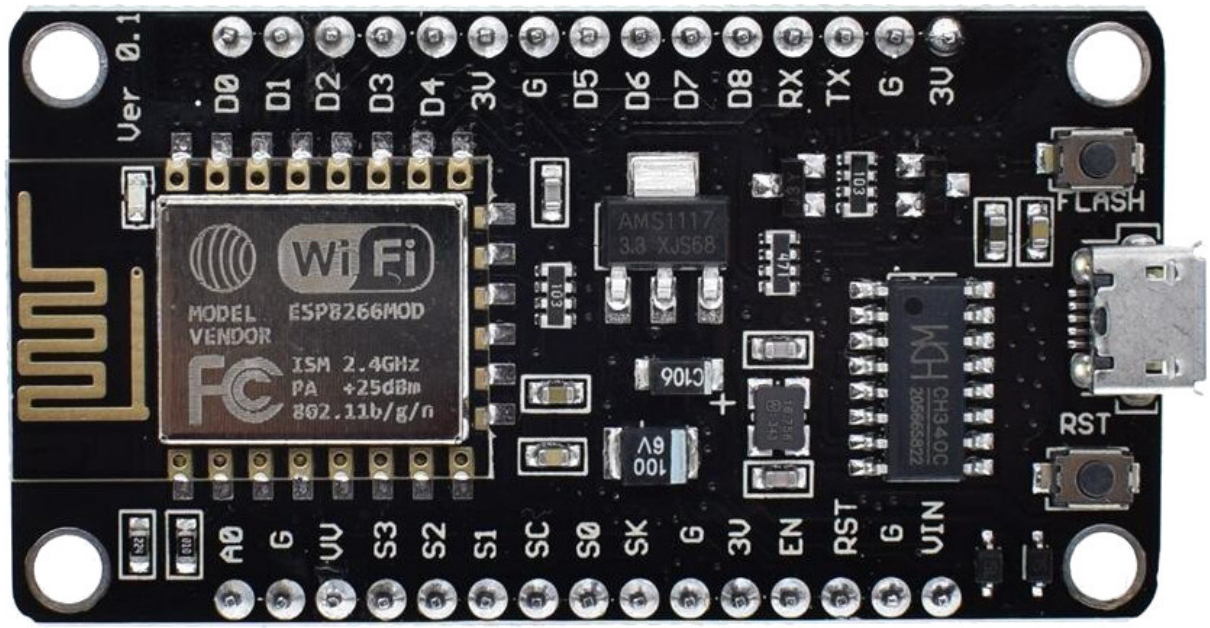
- Visão geral
  - Criada em 1999 pela Tensilica que depois foi comprada pela Cadence Design Systems em 2013
  - Arquitetura RISC de 32 bits extensível e configurável
    - Disponibilizado como código RTL sintetizável (IP)
    - Possibilita que funções ou instruções personalizadas sejam utilizadas de forma nativa no desenvolvimento
    - Redução do tempo de projeto

Introdução

- Xtensa LX
  - Arquitetura de 32 bits @ 400 MHz (< 76 μW/MHz)
  - 64 registradores de propósito geral em janela
  - Personalização através da linguagem *Tensilica Instruction Extension* (TIE) e de formatos de instrução *Flexible Length Instruction Xtension* (FLIX)

Introdução

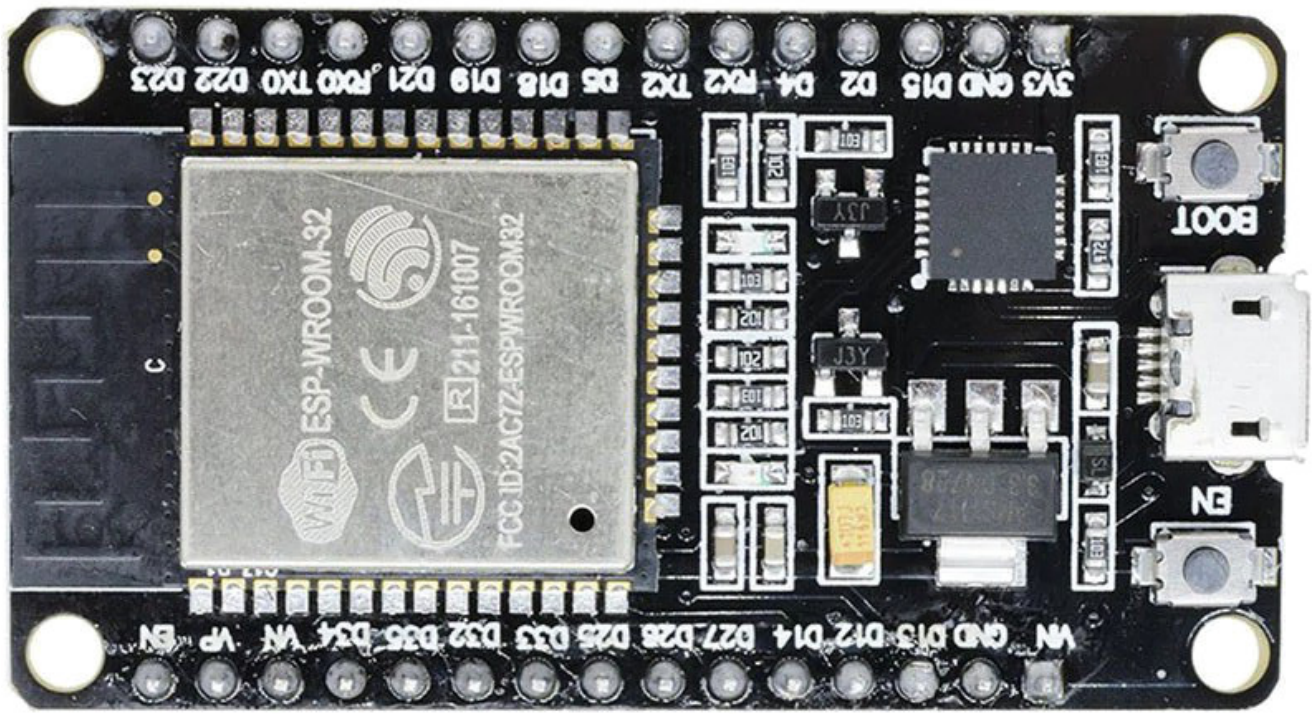
- Placa de desenvolvimento Espressif ESP8266
  - Xtensa LX106 de 32 bits @ 80/160 MHz
  - Custo < US$ 3
  - Temperatura de operação entre -40° C e +125° C
  - Voltagem: 2,5 V até 3,6 V (~ 80 mA)



Fonte: https://www.espressif.com/en/support/download/documents

Introdução

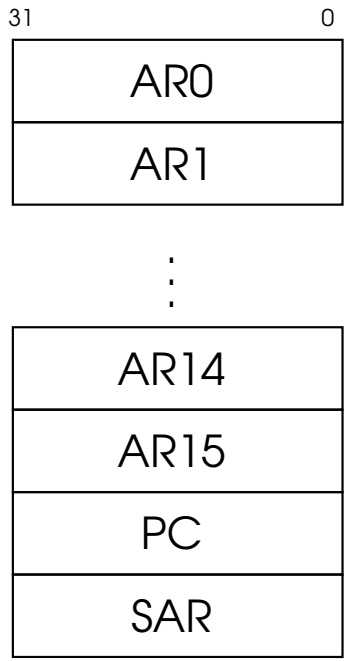
- Placa de desenvolvimento Espressif ESP32
  - Dois núcleos Xtensa LX6 de 32 bits @ 80/240 MHz
  - Custo < US$ 4
  - Temperatura de operação entre -40° C e +85° C
  - Voltagem: 3,0 V até 3,6 V (~ 80 mA)



Fonte: https://www.espressif.com/en/support/download/documents

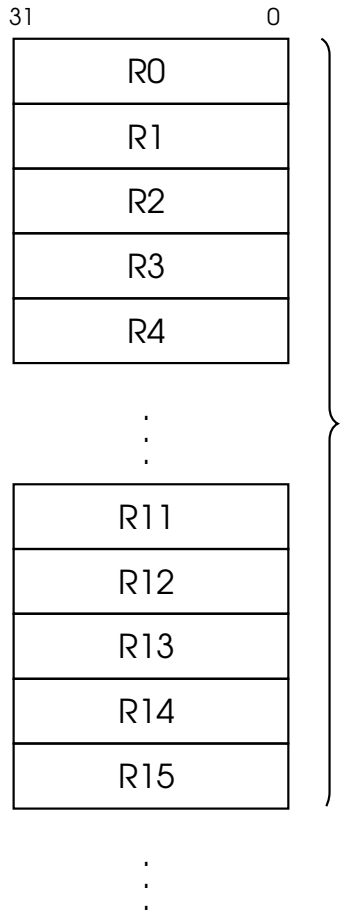
Registradores

- 16 registradores de propósito geral visíveis (AR0 - AR15), contador de programa (PC) e quantidade de deslocamentos (SAR) com 32 bits



Registradores

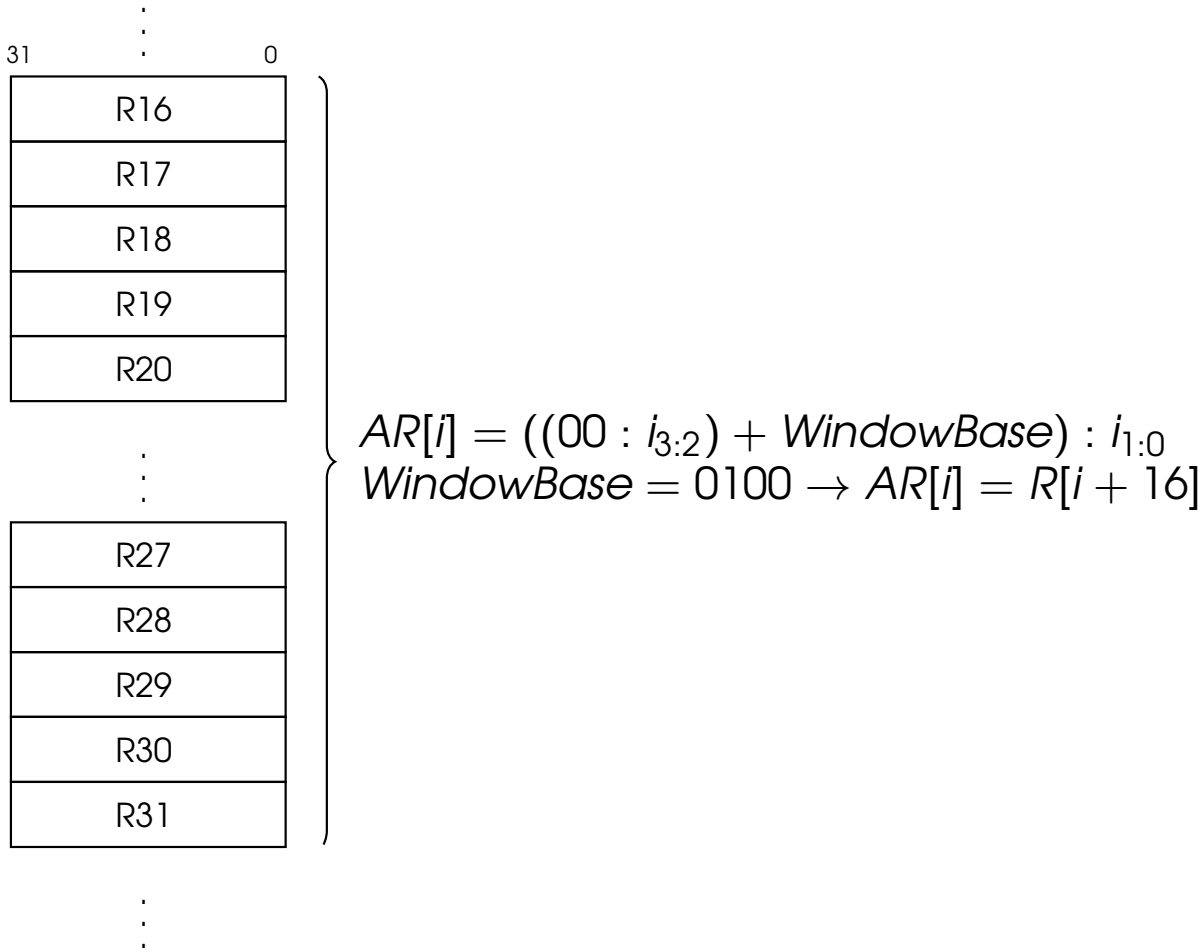
- Janela para rotação dos 64 registradores



$$AR[i] = ((00 : i_{3:2}) + WindowBase) : i_{1:0}$$
$$WindowBase = 0000 \rightarrow AR[i] = R[i]$$

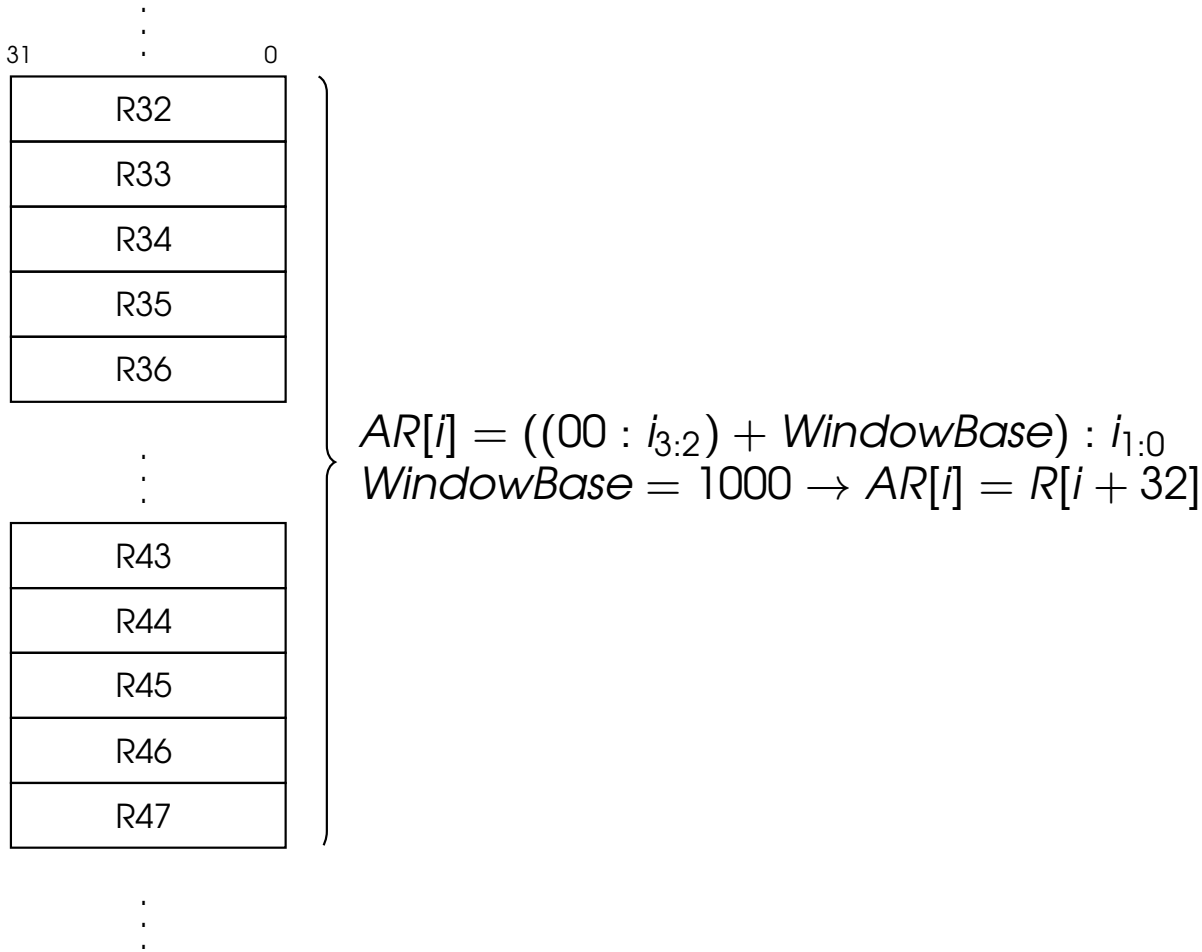
Registradores

- Janela para rotação dos 64 registradores



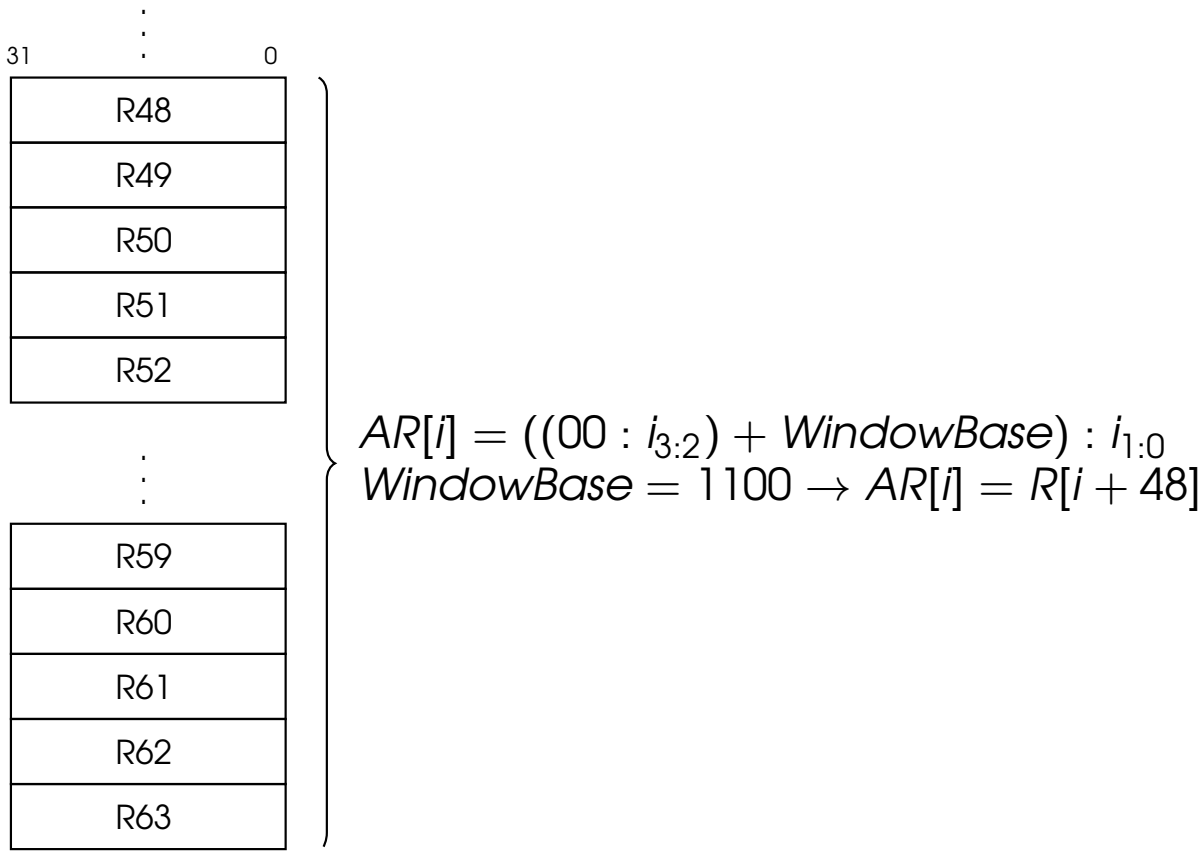
Registradores

- Janela para rotação dos 64 registradores



Registradores

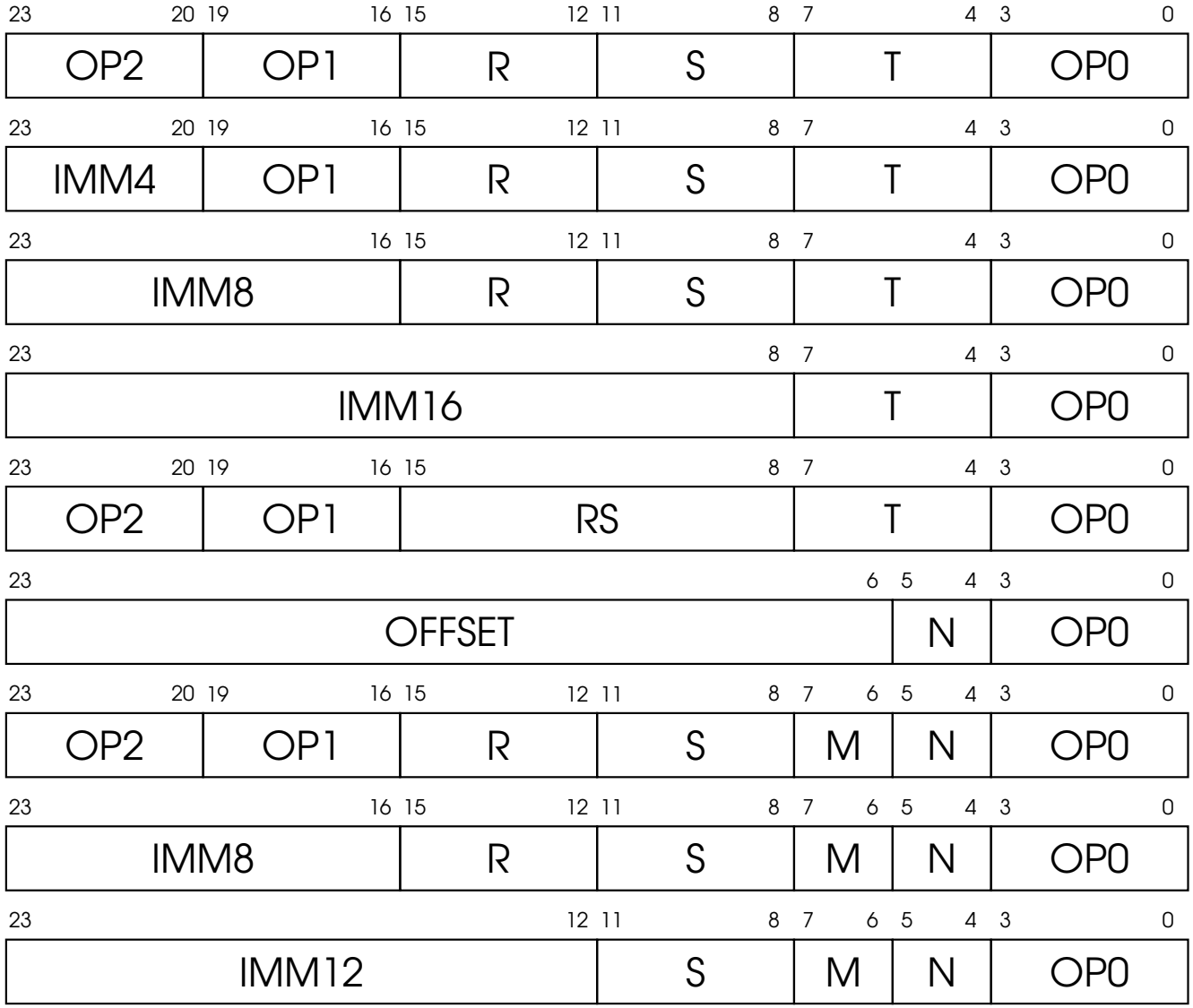
- Janela para rotação dos 64 registradores



O rotacionamento pode ser automático nas chamadas das subrotinas para preservar o contexto

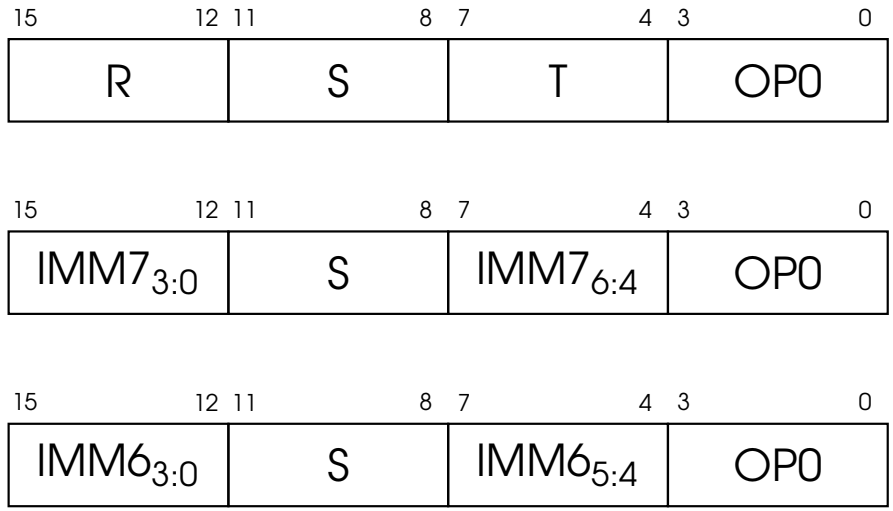
Formatos de instruções

- Instruções com 24 bits
  - RRR, RRI4, RRI8, RI16, RSR, CALL, CALLX, BRI8 e BRI12



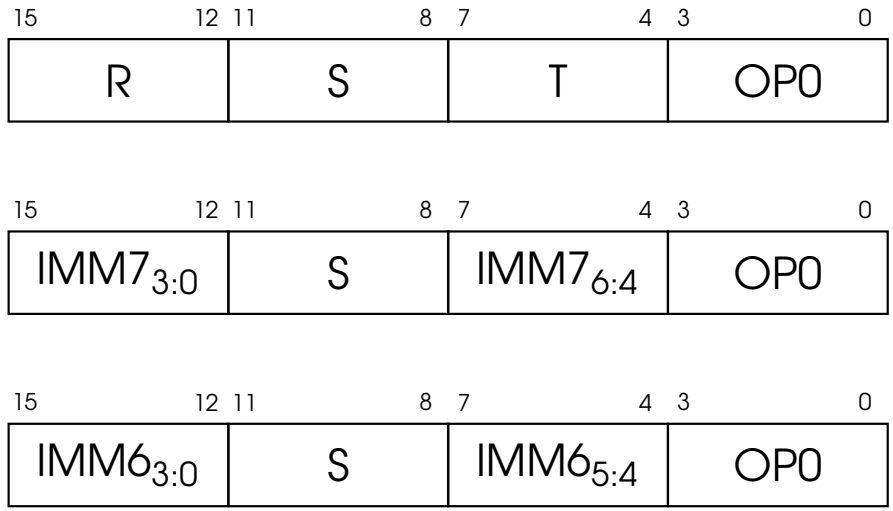
Formatos de instruções

- Instruções com 16 bits
  - RRRN, RI7 e RI6



Formatos de instruções

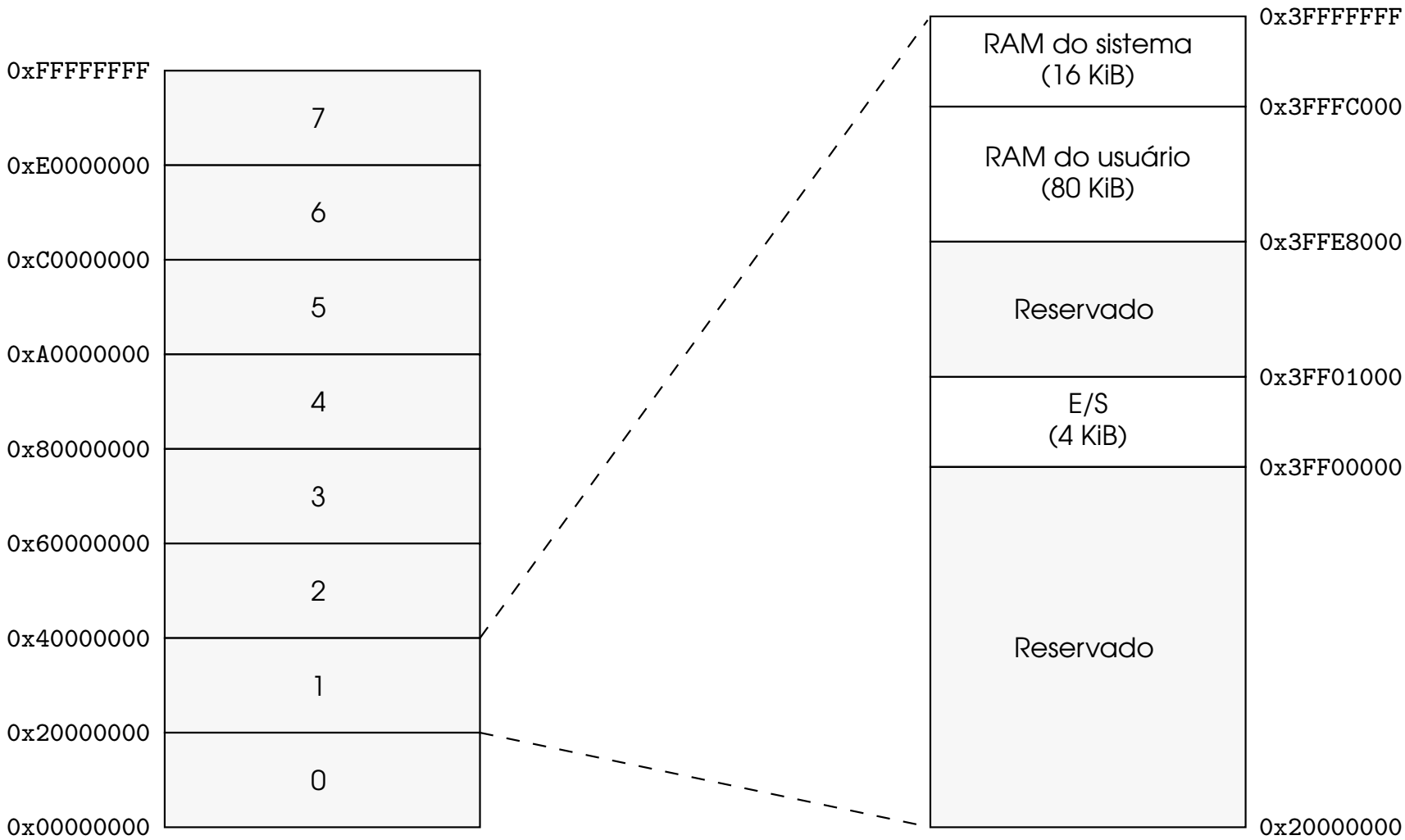
- Instruções com 16 bits
  - RRRN, RI7 e RI6



Também é possível a criação de blocos de instruções com 32 ou 64 bits (FLIX) para execução paralela (VLIW)

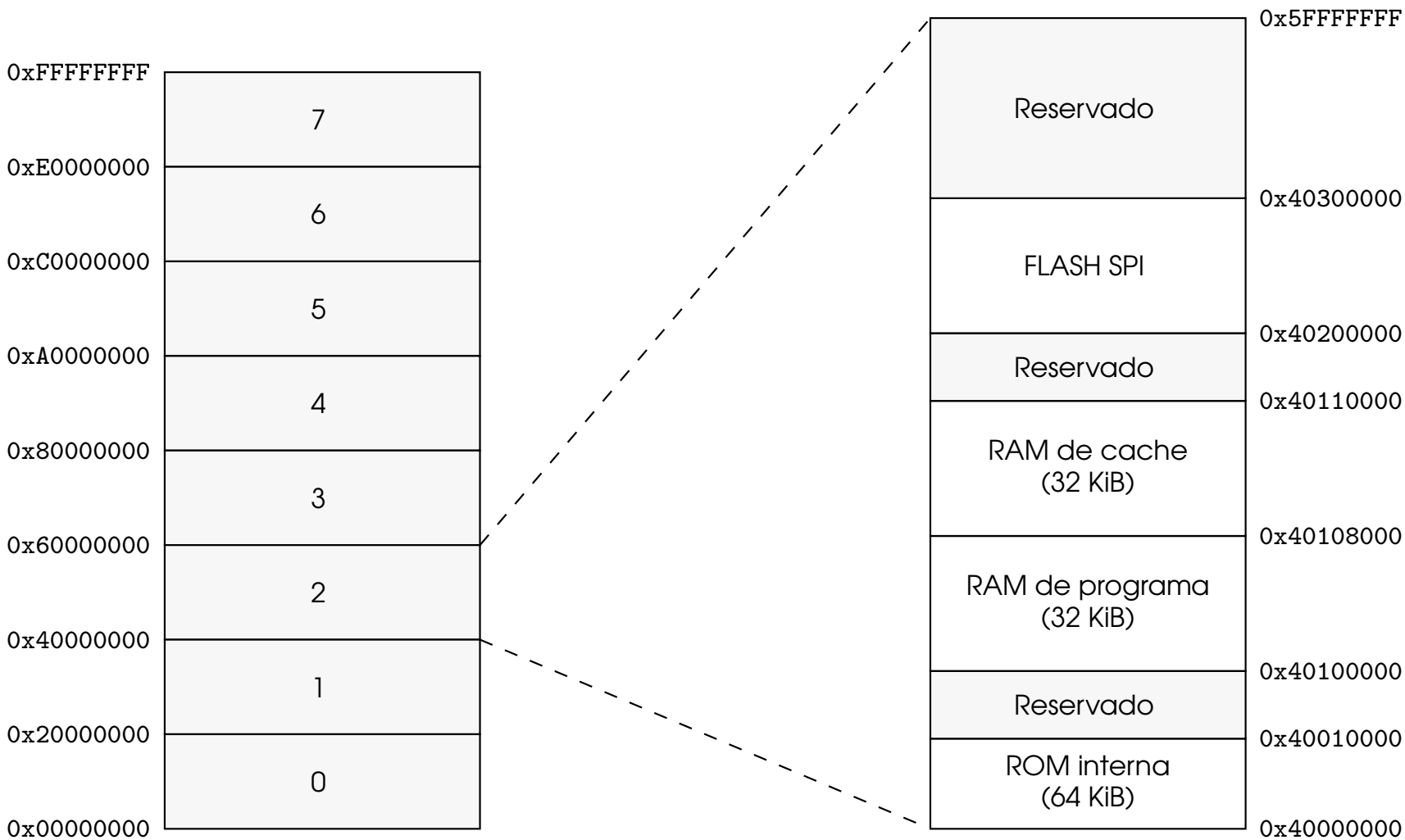
Mapa de memória

- Plataforma ESP8266



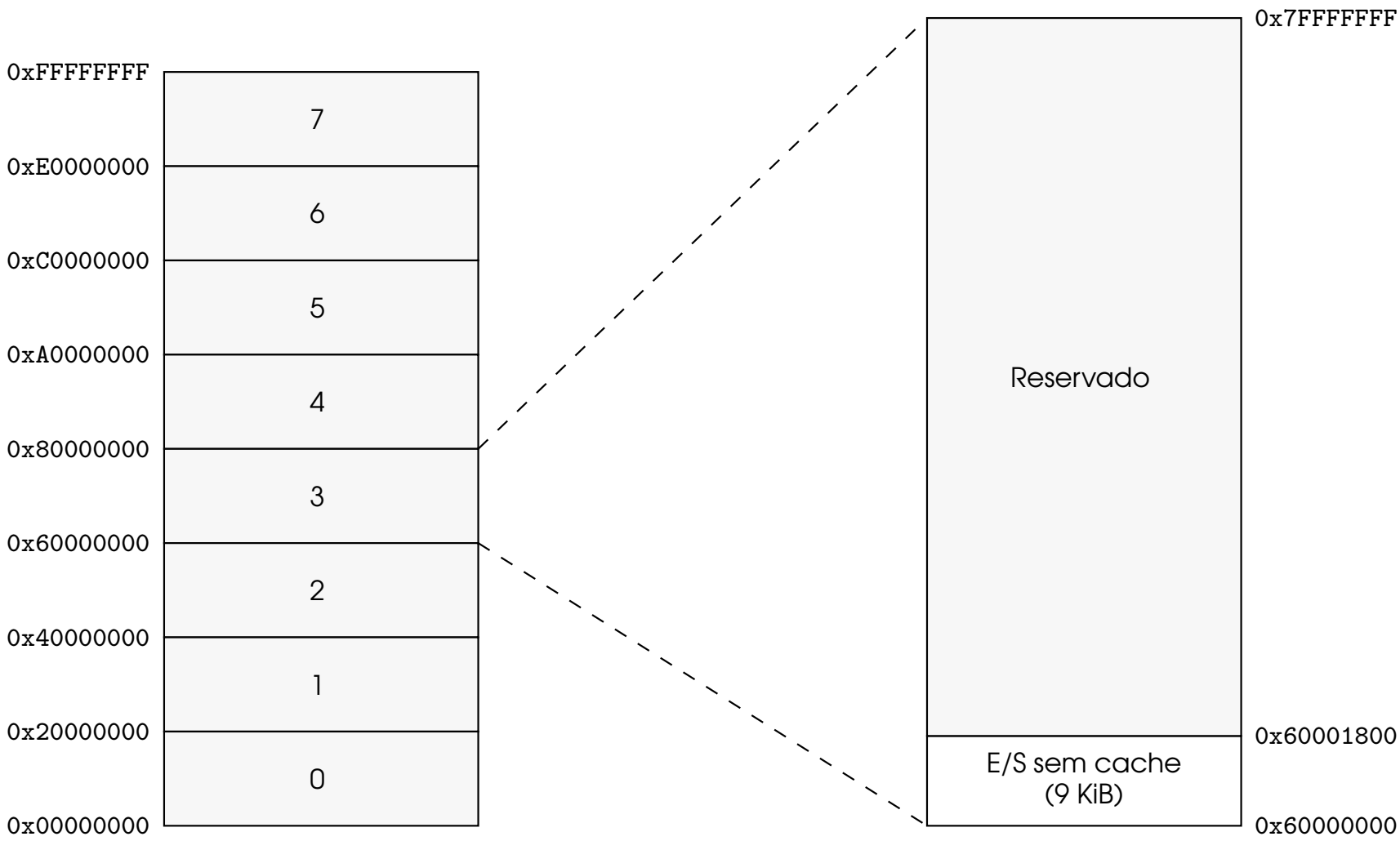
Mapa de memória

- Plataforma ESP8266



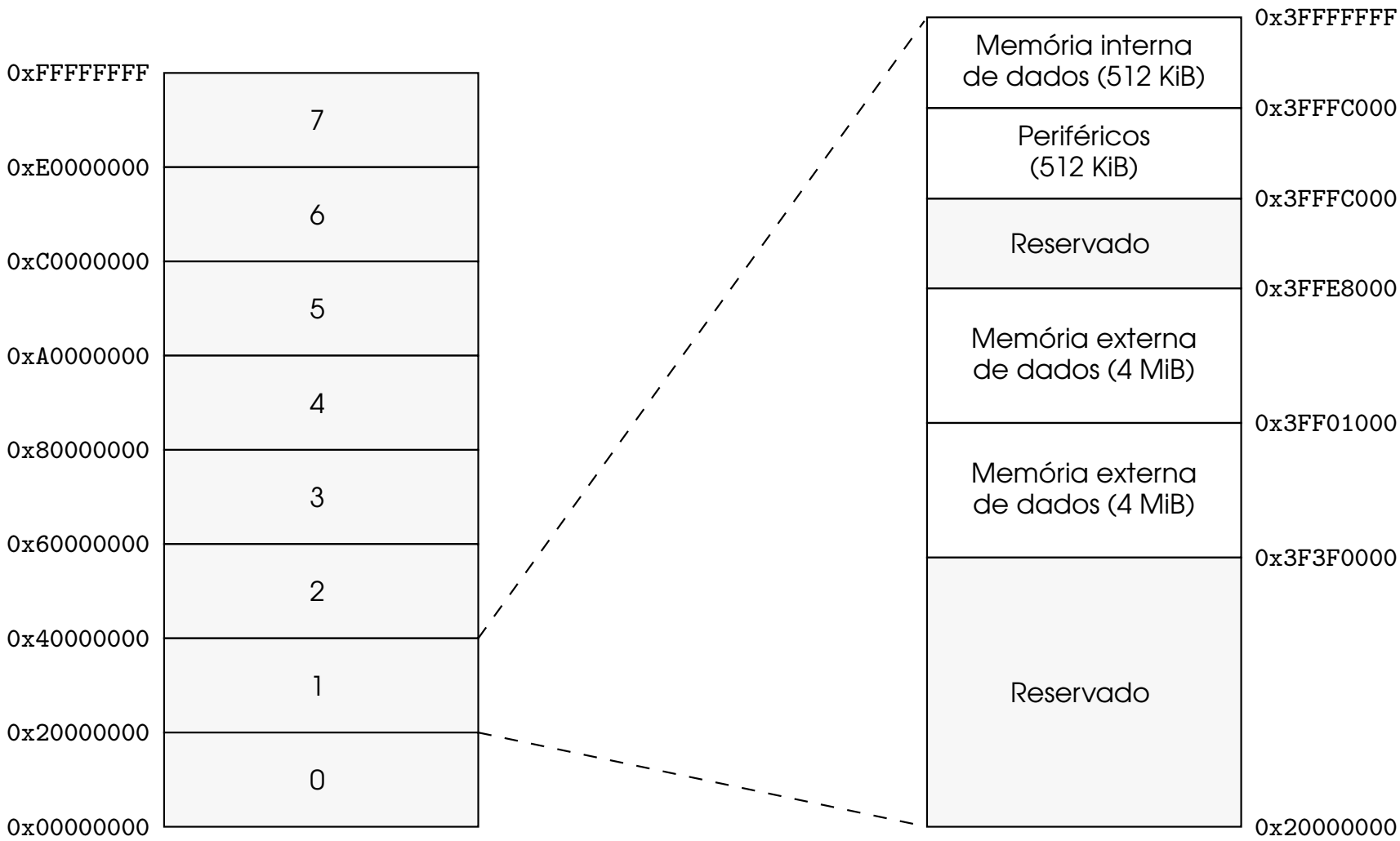
Mapa de memória

- Plataforma ESP8266



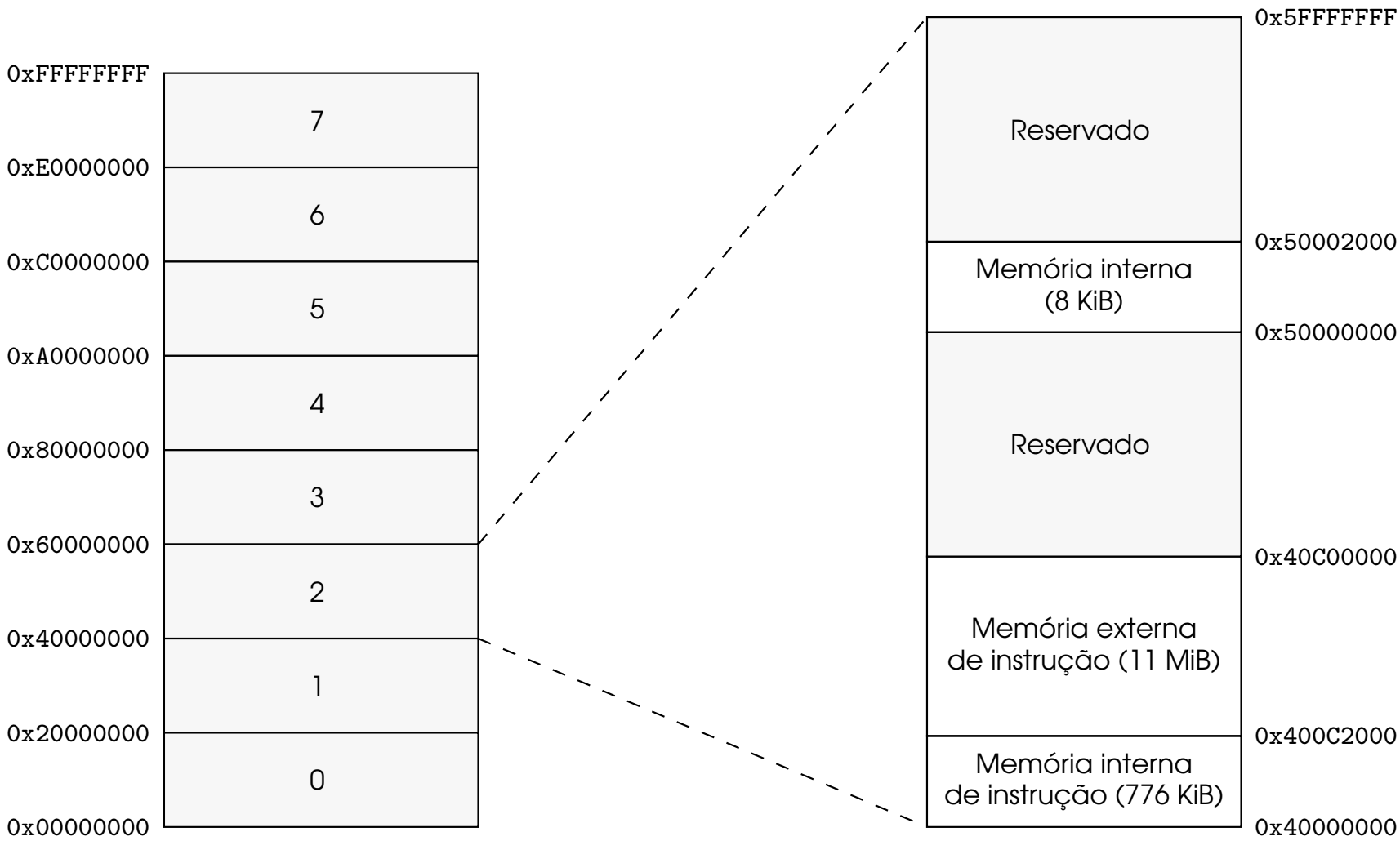
Mapa de memória

- Plataforma ESP32



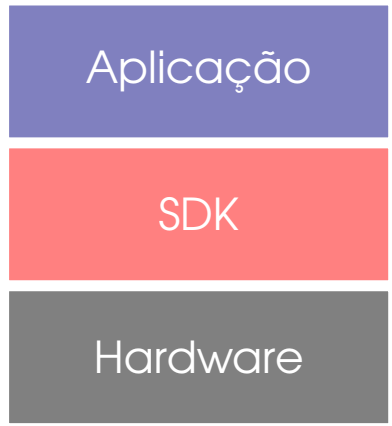
Mapa de memória

- Plataforma ESP32



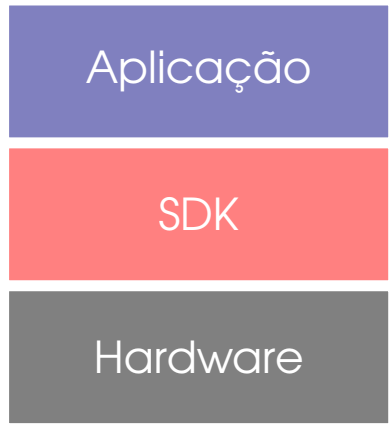
Software Development Kit (SDK)

- Sem sistema operacional (non-OS SDK)



Software Development Kit (SDK)

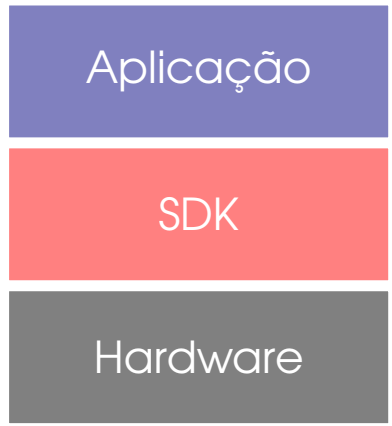
- Sem sistema operacional (non-OS SDK)



  - Disponibiliza um conjunto de interfaces de programação (API) para controle do hardware, como recepção/transmissão por rede sem fio (Wi-Fi) utilizando a pilha de comunicação TCP/IP

Software Development Kit (SDK)

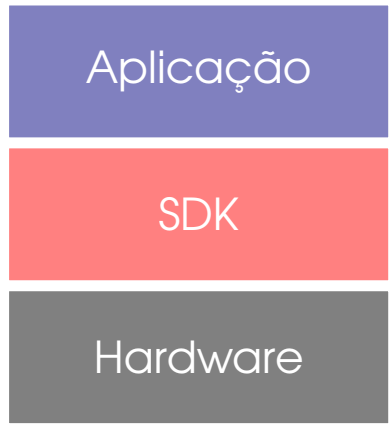
- Sem sistema operacional (non-OS SDK)



  - Disponibiliza um conjunto de interfaces de programação (API) para controle do hardware, como recepção/transmissão por rede sem fio (Wi-Fi) utilizando a pilha de comunicação TCP/IP
  - Reduz a quantidade de memória necessária para o funcionamento do sistema por não usar SO

Software Development Kit (SDK)

- Sem sistema operacional (non-OS SDK)

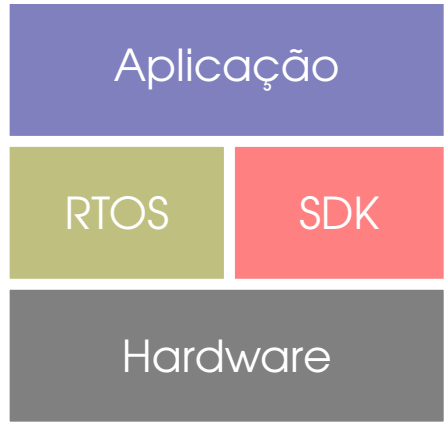


  - Disponibiliza um conjunto de interfaces de programação (API) para controle do hardware, como recepção/transmissão por rede sem fio (Wi-Fi) utilizando a pilha de comunicação TCP/IP
  - Reduz a quantidade de memória necessária para o funcionamento do sistema por não usar SO

https://www.espressif.com/sites/default/files/documentation/2c-esp8266_non_os_sdk_api_reference_en.pdf

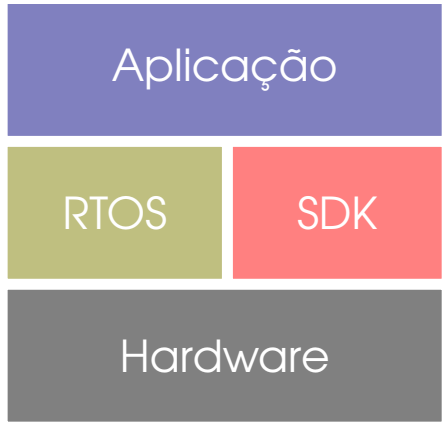
Software Development Kit (SDK)

- Com sistema operacional de tempo real (RTOS SDK)



Software Development Kit (SDK)

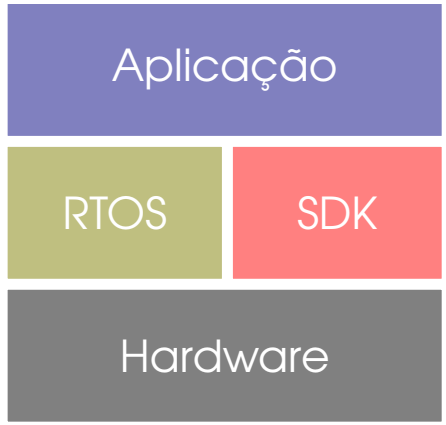
- Com sistema operacional de tempo real (RTOS SDK)



  - O sistema operacional de tempo real (RTOS) possibilita o escalonamento e preempção de múltiplas tarefas, utilizando interfaces padronizadas para realizar o gerenciamento de memória, de temporização, de mensagens entre tarefas e sincronização

Software Development Kit (SDK)

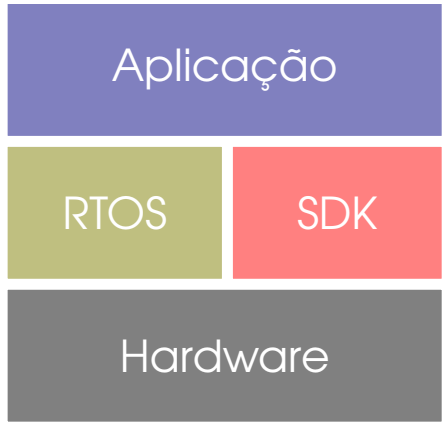
- Com sistema operacional de tempo real (RTOS SDK)



  - O sistema operacional de tempo real (RTOS) possibilita o escalonamento e preempção de múltiplas tarefas, utilizando interfaces padronizadas para realizar o gerenciamento de memória, de temporização, de mensagens entre tarefas e sincronização
  - A utilização do RTOS causa um impacto negativo no tempo de execução e na utilização de memória

Software Development Kit (SDK)

- Com sistema operacional de tempo real (RTOS SDK)



  - O sistema operacional de tempo real (RTOS) possibilita o escalonamento e preempção de múltiplas tarefas, utilizando interfaces padronizadas para realizar o gerenciamento de memória, de temporização, de mensagens entre tarefas e sincronização
  - A utilização do RTOS causa um impacto negativo no tempo de execução e na utilização de memória

https://docs.espressif.com/projects/esp8266-rtos-sdk/en/latest/api-reference/wifi/esp_wifi.html

Desenvolvimento (Non-OS)

- Ligando/desligando LED + Busca por redes sem fio

```
1   // Biblitecas do SDK
2   #include "osapi.h"
3   #include "user_interface.h"
4   // Timer do blink
5   static os_timer_t blink_timer;
6   // Procedimento do blink
7   void ICACHE_FLASH_ATTR blink(void* arg) {
8       // Invertendo valor do pino 2
9       GPIO_OUTPUT_SET(2, !GPIO_INPUT_GET(2));
10  }
... ...
```

Desenvolvimento (Non-OS)

- Ligando/desligando LED + Busca por redes sem fio

```
1   // Biblitecas do SDK
2   #include "osapi.h"
3   #include "user_interface.h"
4   // Timer do blink
5   static os_timer_t blink_timer;
6   // Procedimento do blink
7   void ICACHE_FLASH_ATTR blink(void* arg) {
8       // Invertendo valor do pino 2
9       GPIO_OUTPUT_SET(2, !GPIO_INPUT_GET(2));
10  }
... ...
```

Armazena a função na RAM
para otimizar o desempenho

Desenvolvimento (Non-OS)

- Ligando/desligando LED + Busca por redes sem fio

```
 1  // Bibliotecas do SDK
 2  #include "osapi.h"
...
43  // Procedimento de inicializacao
44  void ICACHE_FLASH_ATTR user_init() {
45      // Iniciando Wi-Fi em modo estacao
46      wifi_set_opmode(STATION_MODE);
47      // Inicializando GPIO
48      gpio_init();
49      // Configurando pinos 2 e 4 como GPIO
50      PIN_FUNC_SELECT(PERIPHS_IO_MUX_GPIO2_U, FUNC_GPIO2);
51      PIN_FUNC_SELECT(PERIPHS_IO_MUX_GPIO4_U, FUNC_GPIO4);
52      // Setando pino 4 como entrada
53      gpio_output_set(0, 0, 0, BIT4);
54      // Ajustando gatilho para nível alto no pino 4
55      gpio_pin_intr_state_set(GPIO_ID_PIN(4),
            GPIO_PIN_INTR_POSEDGE);
...
```

Desenvolvimento (Non-OS)

- Ligando/desligando LED + Busca por redes sem fio

```
1   // Biblitecas do SDK
2   #include "osapi.h"
... ...
56      // Associando handler ao pino 4
57      ETS_GPIO_INTR_ATTACH(wifi_scan, NULL);
58      // Habilitando interrupcao
59      ETS_GPIO_INTR_ENABLE();
60      // Desarmando o timer do blink
61      os_timer_disarm(&blink_timer);
62      // Setando o callback do blink com argumento NULL
63      os_timer_setfn(&blink_timer,
            (os_timer_func_t*)(blink), NULL);
64      // Armando timer para 500 ms com repeticao
65      os_timer_arm(&blink_timer, 500, 1);
66  }
```

Desenvolvimento (Non-OS)

- Ligando/desligando LED + Busca por redes sem fio

```
1   // Biblitecas do SDK
2   #include "osapi.h"
... ...
56      // Associando handler ao pino 4
57      ETS_GPIO_INTR_ATTACH(wifi_scan, NULL);
58      // Habilitando interrupcao
59      ETS_GPIO_INTR_ENABLE();
60      // Desarmando o timer do blink
61      os_timer_disarm(&blink_timer);
62      // Setando o callback do blink com argumento NULL
63      os_timer_setfn(&blink_timer,
            (os_timer_func_t*)(blink), NULL);
64      // Armando timer para 500 ms com repeticao
65      os_timer_arm(&blink_timer, 500, 1);
66  }
```

Associação de funções de *callback*

Desenvolvimento (Non-OS)

- Ligando/desligando LED + Busca por redes sem fio

```
1   // Biblitecas do SDK
2   #include "osapi.h"
... ...
29  // Procedimento para escanear redes sem fio
30  void ICACHE_FLASH_ATTR wifi_scan(void* arg) {
31      // Checando status do pino 4
32      if(GPIO_REG_READ(GPIO_STATUS_ADDRESS) & BIT(4)) {
33          // Desabilitanto interrupcao no pino 4
34          gpio_pin_intr_state_set(GPIO_ID_PIN(4),
                GPIO_PIN_INTR_DISABLE);
35          // Buscando pontos de acesso
36          wifi_station_scan(NULL, wifi_list);
... ...
```

Desenvolvimento (Non-OS)

- Ligando/desligando LED + Busca por redes sem fio

```
1   // Biblitecas do SDK
2   #include "osapi.h"
... ...
29  // Procedimento para escanear redes sem fio
30  void ICACHE_FLASH_ATTR wifi_scan(void* arg) {
31      // Checando status do pino 4
32      if(GPIO_REG_READ(GPIO_STATUS_ADDRESS) & BIT(4)) {
33          // Desabilitanto interrupcao no pino 4
34          gpio_pin_intr_state_set(GPIO_ID_PIN(4),
                GPIO_PIN_INTR_DISABLE);
35          // Buscando pontos de acesso
36          wifi_station_scan(NULL, wifi_list);
... ...
```

Após o escaneamento das redes,
é chamada a função de *callback* wifi_list

Desenvolvimento (Non-OS)

- Ligando/desligando LED + Busca por redes sem fio

```
1   // Biblitecas do SDK
2   #include "osapi.h"
... ...
37          // Limpando flag de interrupcao no pino 4
38          GPIO_REG_WRITE(GPIO_STATUS_W1TC_ADDRESS,
                GPIO_REG_READ(GPIO_STATUS_ADDRESS) &
                BIT(4));
39          // Habilitando interrupcao no pino 4
40          gpio_pin_intr_state_set(GPIO_ID_PIN(4),
                GPIO_PIN_INTR_POSEDGE);
41      }
42  }
... ...
```

Desenvolvimento (Non-OS)

- Ligando/desligando LED + Busca por redes sem fio

```
 1  // Biblitecas do SDK
 2  #include "osapi.h"
...
11  // Procedimento para listar redes sem fio
12  void ICACHE_FLASH_ATTR wifi_list(void* arg, STATUS status) {
13      // Checagem de status
14      if(status == OK) {
15          // Separador de texto
16          os_printf("---------------------------------\n");
17          // Iterando na lista
18          struct bss_info* lista = (struct bss_info *)(arg);
19          while(lista != NULL) {
20              // Imprimindo dados da rede sem fio
21              os_printf("%-20s CANAL %02u (%02d)\n",
                    lista->ssid, lista->channel, lista->rssi);
22              // Busca do proximo elemento da lista
23              lista = STAILQ_NEXT(lista, next);
24          }
25      }
26      // Mensagem de erro
27      else os_printf("Erro na busca por redes sem fio!\n");
28  }
...
```

Desenvolvimento (RTOS)

- Ligando/desligando LED + Busca por redes sem fio

```
1   // Biblitecas do SDK
2   #include "esp_common.h"
3   #include "gpio.h"
... ...
71  // Procedimento de inicializacao
72  void user_init() {
73      // Criando tarefa do blink
74      xTaskCreate(blink, "blink", 256, NULL, 2, NULL);
75      // Criando tarefa do wifi
76      xTaskCreate(wifi, "wifi", 256, NULL, 1, NULL);
77  }
```

Desenvolvimento (RTOS)

- Ligando/desligando LED + Busca por redes sem fio

```
1   // Biblitecas do SDK
2   #include "esp_common.h"
3   #include "gpio.h"
... ...
71  // Procedimento de inicializacao
72  void user_init() {
73      // Criando tarefa do blink
74      xTaskCreate(blink, "blink", 256, NULL, 2, NULL);
75      // Criando tarefa do wifi
76      xTaskCreate(wifi, "wifi", 256, NULL, 1, NULL);
77  }
```

https://www.freertos.org/a00125.html

Desenvolvimento (RTOS)

- Ligando/desligando LED + Busca por redes sem fio

```
1   // Biblitecas do SDK
2   #include "esp_common.h"
... ...
4   // Tarefa do blink
5   void ICACHE_FLASH_ATTR blink(void* pvParameters) {
6       // Configurando pino 2 como GPIO
7       PIN_FUNC_SELECT(PERIPHS_IO_MUX_GPIO2_U,
            FUNC_GPIO2);
8       // Laço infinito
9       while(1) {
10          // Invertendo valor do pino 2
11          GPIO_OUTPUT_SET(2, !GPIO_INPUT_GET(2));
12          // Delay de 500 ms
13          vTaskDelay(500 / portTICK_RATE_MS);
14      }
15      // Finalizando tarefa
16      vTaskDelete(NULL);
17  }
```

Desenvolvimento (RTOS)

- Ligando/desligando LED + Busca por redes sem fio

```
1   // Biblitecas do SDK
2   #include "esp_common.h"
... ...
50  // Tarefa do wifi
51  void wifi(void* pvParameters) {
52      // Mensagem da tarefa
53      printf("WIFI START!\n");
54      // Iniciando Wi-Fi em modo estacao
55      wifi_set_opmode(STATION_MODE);
... ...
64      // Habilitando interrupcao
65      _xt_isr_unmask(1 << ETS_GPIO_INUM);
66      // Mensagem da tarefa
67      printf("WIFI END!\n");
68      // Finalizando tarefa
69      vTaskDelete(NULL);
70  }
```

Execução

- Monitor serial

Execução

- Monitor serial

```
$ miniterm.py /dev/ttyUSB0 76800
```

Execução

- Monitor serial

```
$ miniterm.py /dev/ttyUSB0 76800
--- Miniterm on /dev/ttyUSB0  76800,8,N,1 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---

 ets Jan  8 2013,rst cause:2, boot mode:(3,6)

load 0x40100000, len 31360, room 16
tail 0
chksum 0x22
load 0x3ffe8000, len 2128, room 8
tail 8 chksum 0x58
load 0x3ffe8850, len 624, room 0
tail 0
chksum 0x1d
csum 0x1d
OS SDK ver: 1.5.0-dev(caff253) compiled @ Oct 23 2017 17:42:20
phy ver: 1055_1, pp ver: 10.7

rf cal sector: 1019
```

Execução

- Monitor serial

```
tail 0
chksum 0x22
load 0x3ffe8000, len 2128, room 8
tail 8 chksum 0x58
load 0x3ffe8850, len 624, room 0
tail 0
chksum 0x1d
csum 0x1d
OS SDK ver: 1.5.0-dev(caff253) compiled @ Oct 23 2017 17:42:20
phy ver: 1055_1, pp ver: 10.7

rf cal sector: 1019
tcpip_task_hdl : 3ffef6a0, prio:10,stack:512
idle_task_hdl : 3ffef740,prio:0, stack:384
tim_task_hdl : 3fff1ef8, prio:2,stack:512
mode : sta(80:7d:3a:6e:38:ad)
add if0
WIFI START!
WIFI END!
```

Execução

- Monitor serial

```
chksum 0x22
load 0x3ffe8000, len 2128, room 8
tail 8 chksum 0x58
load 0x3ffe8850, len 624, room 0
tail 0
chksum 0x1d
csum 0x1d
OS SDK ver: 1.5.0-dev(caff253) compiled @ Oct 23 2017 17:42:20
phy ver: 1055_1, pp ver: 10.7

rf cal sector: 1019
tcpip_task_hdl : 3ffef6a0, prio:10,stack:512
idle_task_hdl : 3ffef740,prio:0, stack:384
tim_task_hdl : 3fff1ef8, prio:2,stack:512
mode : sta(80:7d:3a:6e:38:ad)
add if0
WIFI START!
WIFI END!
scandone
```

Interrupção externa (botão pressionado)

Execução

- Monitor serial

```
phy ver: 1055_1, pp ver: 10.7

rf cal sector: 1019
tcpip_task_hdl : 3ffef6a0, prio:10,stack:512
idle_task_hdl : 3ffef740,prio:0, stack:384
tim_task_hdl : 3fff1ef8, prio:2,stack:512
mode : sta(80:7d:3a:6e:38:ad)
add if0
WIFI START!
WIFI END!
scandone
-----------------------------------
AAA                    CANAL 01 (-42)
BBBBBBBBBB             CANAL 01 (-90)
CCCCCCCCCCC            CANAL 05 (-80)
DDDDDDDDDDDDDD         CANAL 06 (-91)
EEEEEEEEEEEE           CANAL 09 (-81)
FFFFFFFFFFFFFF         CANAL 09 (-80)
GGGGGGGGGG             CANAL 11 (-81)
HHHHHHHH               CANAL 11 (-84)
```

Execução

- Monitor serial

```
rf cal sector: 1019
tcpip_task_hdl : 3ffef6a0, prio:10,stack:512
idle_task_hdl : 3ffef740,prio:0, stack:384
tim_task_hdl : 3fff1ef8, prio:2,stack:512
mode : sta(80:7d:3a:6e:38:ad)
add if0
WIFI START!
WIFI END!
scandone
-------------------------------------
AAA                  CANAL 01 (-42)
BBBBBBBBBBB          CANAL 01 (-90)
CCCCCCCCCCCC         CANAL 05 (-80)
DDDDDDDDDDDDDD       CANAL 06 (-91)
EEEEEEEEEEEEE        CANAL 09 (-81)
FFFFFFFFFFFFFF       CANAL 09 (-80)
GGGGGGGGGG           CANAL 11 (-81)
HHHHHHHHH            CANAL 11 (-84)

--- exit ---
```

Exercício

- Estude e reproduza os experimentos vistos nesta aula
  - Analise os manuais técnicos (*datasheets*) das famílias de microcontroladores ESP8266 e ESP32
  - Verifique o impacto da utilização do Non-OS versus RTOS em termos de desempenho e de memória
  - Utilizando os *frameworks* vistos, realize a conexão com uma rede sem fio e crie um servidor que retorna mensagens de texto enviadas com protocolo TCP