Inicializando a Pilha:

O *Stack Pointer* é pré-decrementado e pós-incrementado:

Armazenar um endereço na pilha: o *SP* é inicialmente decrementado e após o endereço é colocado na pilha.

Retirar um endereço da pilha: o endereço é retirado da pilha e o SP é incrementado.

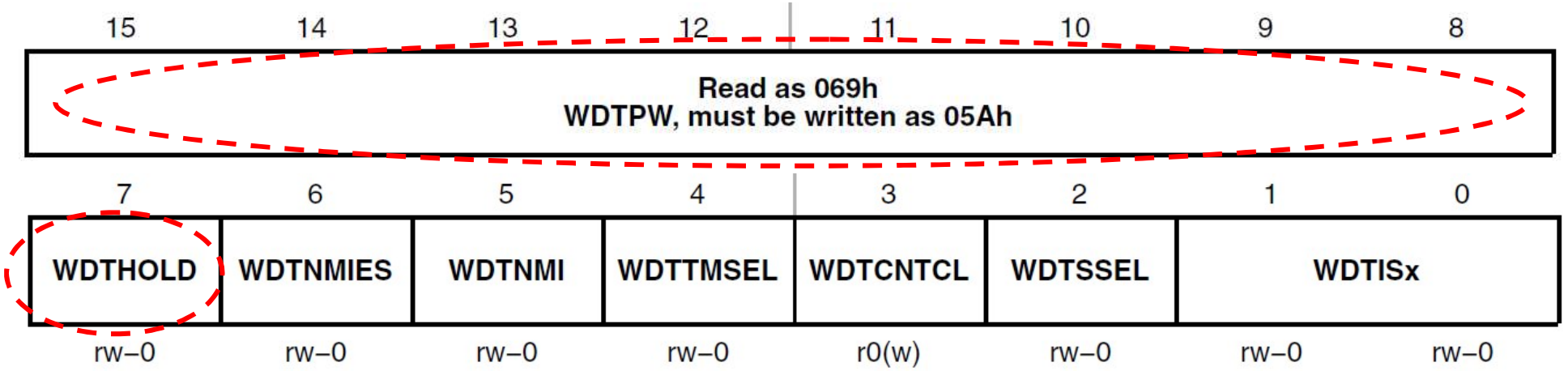
Memória de Dados (RAM)


```
        mov.w   #0x0400, SP          ; inicializar o  Stack Pointer

init:   mov.w   #SFE(CSTACK), SP     ; set up stack
```

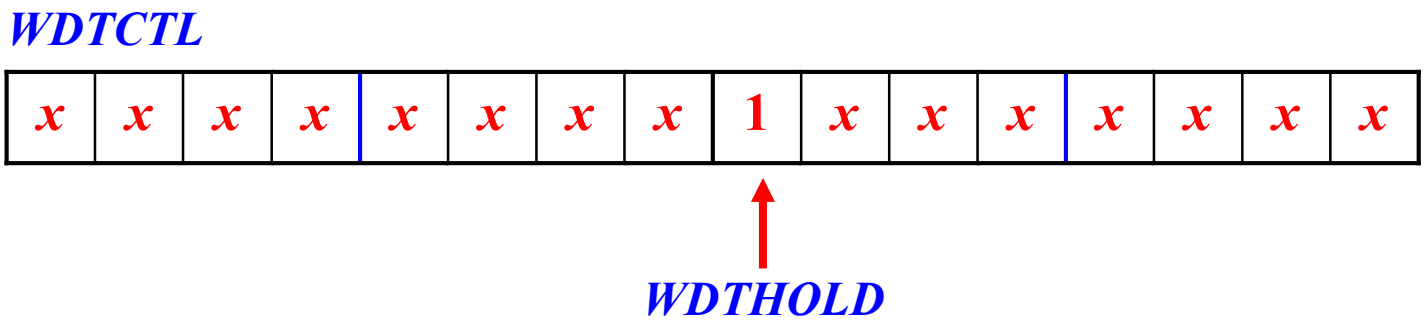
Desativando o WDT:

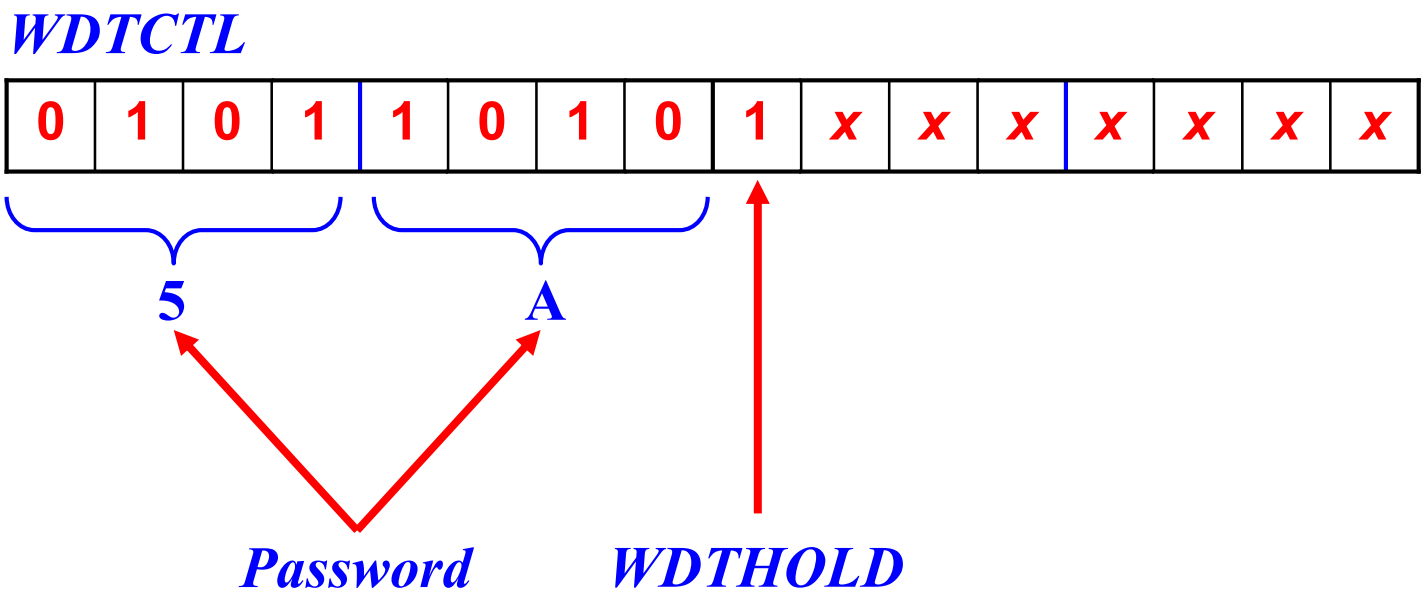
Registrador *WDTCTL*



WDTPW	Bits 15-8	Watchdog timer+ password. Always read as 069h. Must be written as 05Ah, or a PUC will be generated.
WDTHOLD	Bit 7	Watchdog timer+ hold. This bit stops the watchdog timer+. Setting WDTHOLD = 1 when the WDT+ is not in use conserves power. 0 Watchdog timer+ is not stopped 1 Watchdog timer+ is stopped

Desativando o WDT:





Desativando o WDT:

```
#include <msp430.h>        // Definições para o microcontrolador MSP430
```

msp430.h:

```
#elif defined (__MSP430G2553__)
#include "msp430g2553.h"
```

msp430G2553.h:

```
#define WDTCTL_          (0x0120u)  /* Watchdog Timer Control */
DEFW(   WDTCTL           , WDTCTL_)
/* The bit names have been prefixed with "WDT" */
#define WDTIS0           (0x0001u)
#define WDTIS1           (0x0002u)
#define WDTSSEL          (0x0004u)
#define WDTCNTCL         (0x0008u)
#define WDTTMSEL         (0x0010u)
#define WDTNMI           (0x0020u)
#define WDTNMIES         (0x0040u)
#define WDTHOLD          (0x0080u)
#define WDTPW            (0x5A00u)
```

Desativando o WDT:

WDTPW = *0x5A00* = 0101101000000000

+

WDTHOLD = *0x0080* = 0000000010000000

0101101010000000

WDTHOLD

- .
- .
- .

```
mov.w   #WDTPW + WDTHOLD, &WDTCTL       // Desabilita o WDT
```

- .
- .
- .

Exercício:

Mover o valor 0x0220 para o Registrador R4 (R4 ← 0x0220)
Mover o valor 0x0240 para o Registrador R8 (R8 ← 0x0240)
Mover o valor 0x79 para o Registrador R12 (R12 ← 0x79)
Mover o valor 0x8736 para o Registrador R14 (R14 ← 0x8736)

Mover o valor 0x5489 → para o endereço 0x021A (endereçamento absoluto)

R4	02	20
R8	02	40
R12	00	79
R14	87	36

0x0200								
0x0210						5489		
0x0220								

Exercício:

R4	02	20
R8	02	40
R12	00	79
R14	87	36

0x0200								
0x0210						5489		
0x0220								

Definir, na memória de dados(RAM), com início no endereço 0x0200, as variáveis x (byte) e y (word)

Mover o conteúdo do Reg. R12 para a variável x (endereçamento simbólico)

Mover o valor 0x7159 para a variável y (endereçamento simbólico)

Exercício:

R4	02	20
R8	02	40
R12	00	79
R14	87	36

0x0200	79	7159						
0x0210						5489		
0x0220								

Mover o valor 0x9587 para o endereço contido no Reg R4 (endereçamento indexado)
Mover o conteúdo do Reg. R14 para o endereço contido no Reg R8 (endereçamento indexado)

Exercício:

R4	02	20
R8	02	40
R12	00	79
R14	87	36

0x0200	79	7159						
0x0210						5489		
0x0220	9587							
0x0230								
0x0240	8736							
0x0250								

Somar 2 ao Registrador R4
Somar 2 ao Registrador R8

Mover o conteúdo do end. 0x021A para o end. contido no Reg R4 (endereçamento indexado)
Mover o conteúdo da variável y para o end. contido no Reg R8(endereçamento indexado)

Exercício:

R4	02	20
R8	02	40
R12	00	79
R14	87	36

0x0200	79	7159						
0x0210						5489		
0x0220	9587	5489						
0x0230								
0x0240	8736	7159						
0x0250								

Somar a *word* armazenada no endereço 0x0220 com a *word* armazenada no endereço 0x0240, armazenando a soma no endereço 0x0250;

Somar a *word* armazenada no endereço 0x0222 com a *word* armazenada no endereço 0x0242, armazenando a soma no endereço 0x0252;

Obs.: Utilizar endereçamento indireto(origem) e indexado(destino)