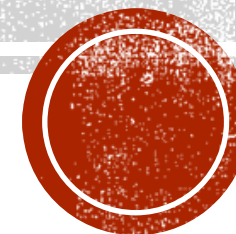


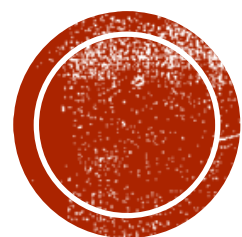
# SISTEMAS MICROPROCESSADOS I

Prof.: João Castelo



CENTRO UNIVERSITÁRIO  
**SENAI CIMATEC**





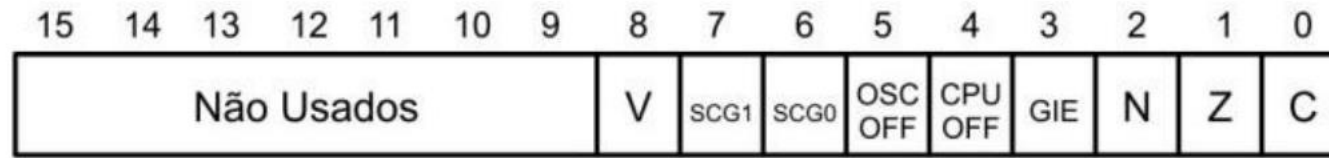
**REVISANDO**



# REGISTRADOR R2/SR/CG1

- O SR possui o propósito de armazenar bits de estado (flags) e de controle da CPU.

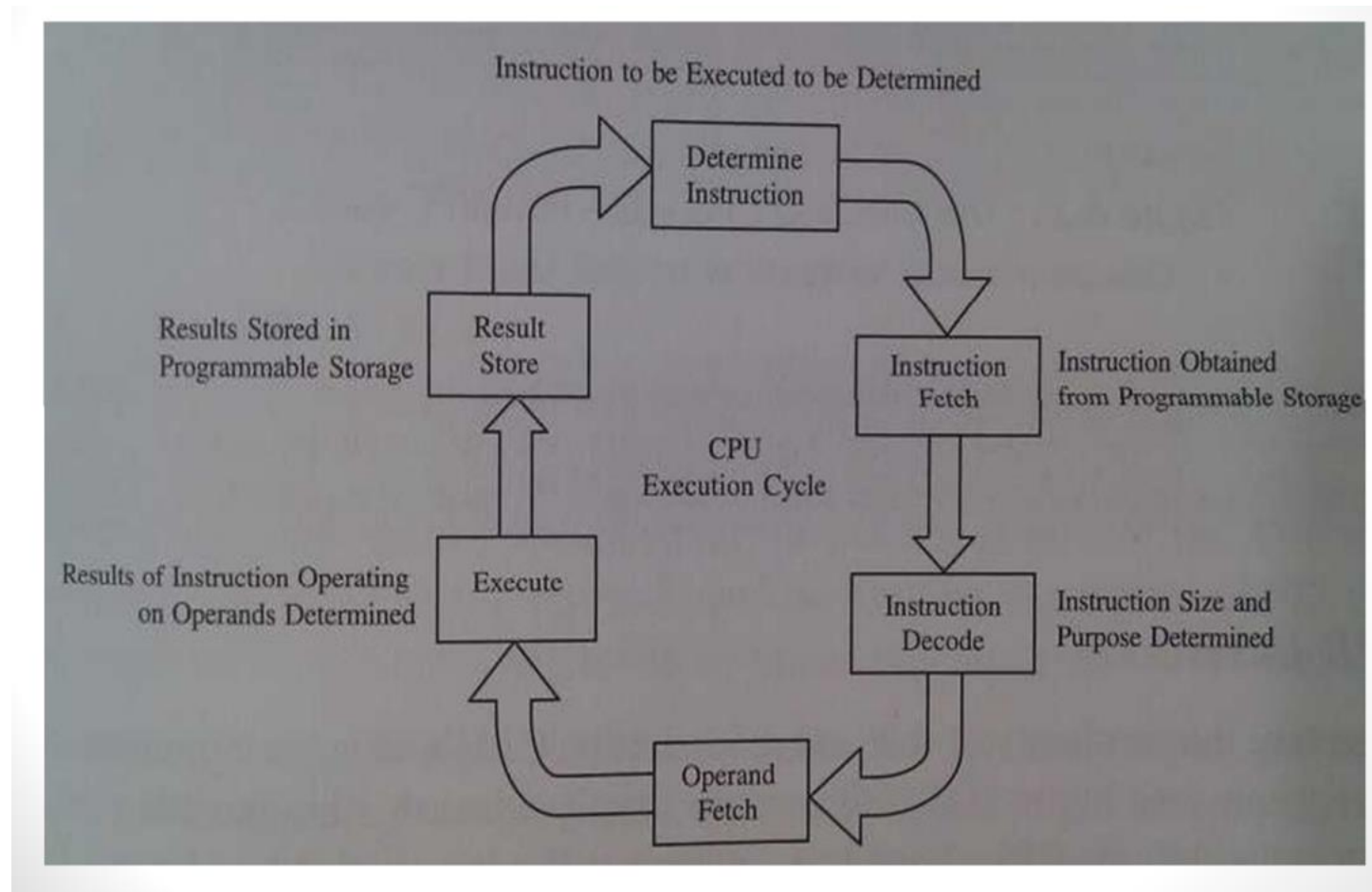
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Não Usados							V	SCG1	SCG0	OSC OFF	CPU OFF	GIE	N	Z	C



- V – Flag de estouro (overflow) – esse bit indica se o resultado da operação envolvendo operando sinalizados ultrapassou o limite de representação da variável (8 bits:  $>127$  ou  $<-128$ ) (16 bits:  $>32767$  ou  $<-32768$ ).
- SCG1 e SCG0 – módulo básico de clock
- OSCOFF – desliga o oscilador quando igual a 1.
- CPUOFF – desliga a CPU quando igual a 1.
- GIE – Bit de controle global de interrupções. Habilita todas as interrupções quando igual a 1 e desabilita todas as interrupções quando igual a 0.
- N – Flag de resultado negativo. Assume valor 1 sempre que uma operação na ULA resulta em um número negativo.
- Z – Flag de zero. Assume valor 1 sempre que o resultado de uma operação na ULA é igual a zero.
- C – Flag de transporte (carry) – Assume valor 1 sempre que uma operação na ULA gera um carry de saída.

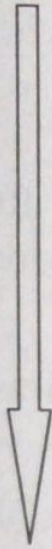
# CICLO DE VIDA – CPU

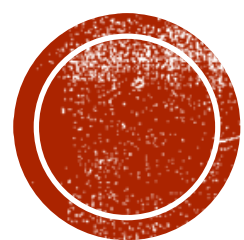
## FETCH – DECODE - EXECUTE





# MSP430 – MODOS DE OPERAÇÃO

Consumo	Modo	CPUOFF	OSCOFF	SCG0	SCG1	MCLK	SMCLK	ACLK	Descrição
Maior	Normal	0	0	0	0	S	S	S	Funcionamento normal, CPU ativa e todos os sinais de <i>clock</i> ativos
	LPM0	1	0	0	0	N	S	S	CPU para e o sinal de <i>clock</i> principal (MCLK) é desativado. Os sinais de <i>clock</i> auxiliares (SMCLK e ACLK) permanecem ativos
	LPM1	1	0	1	0	N	S	S	Idem ao LPM0, mas o DCO é desativado. O gerador DC do DCO é desativado caso não esteja sendo utilizado para gerar o SMCLK ou o ACLK
	LPM2	1	0	0	1	N	N	S	Idem ao LPM1, mas o sinal SMCLK é desativado
	LPM3	1	0	1	1	N	N	S	Idem ao LPM2, mas o gerador DC do DCO é desativado
Menor	LPM4	1	1	1	1	N	N	N	A CPU e todos os sinais de <i>clock</i> são desativados



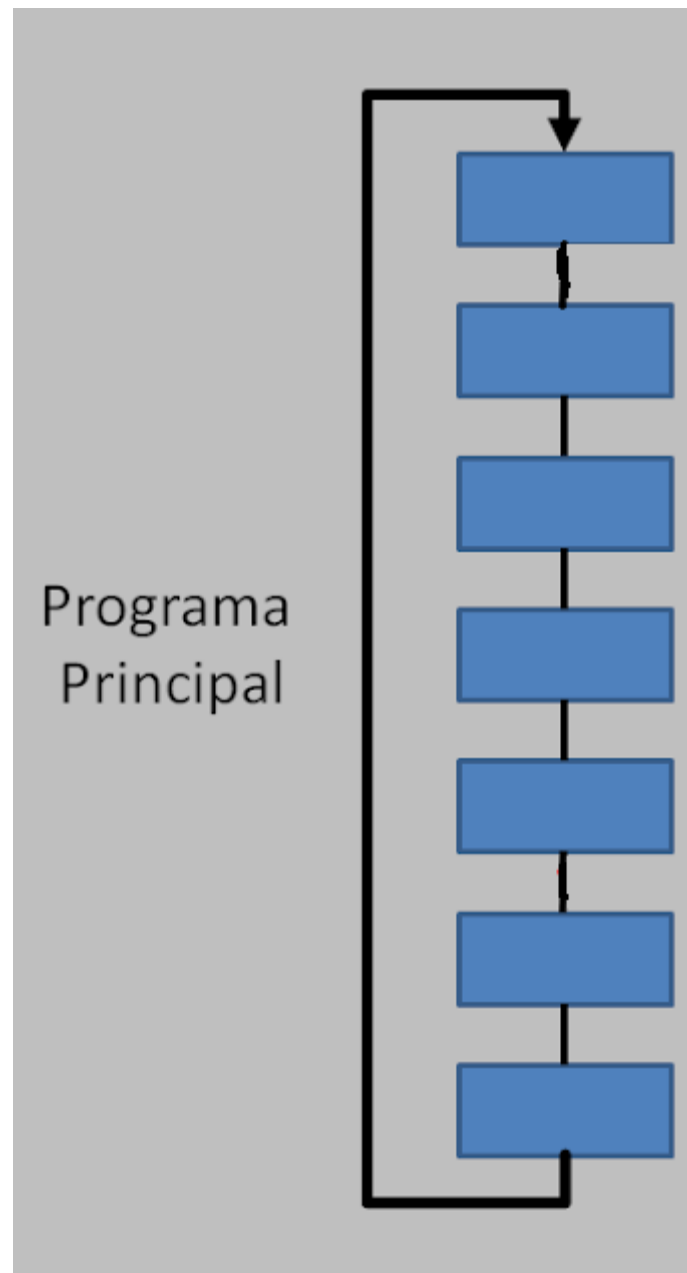
**VAMOS LÁ...**



# INTERRUPÇÃO

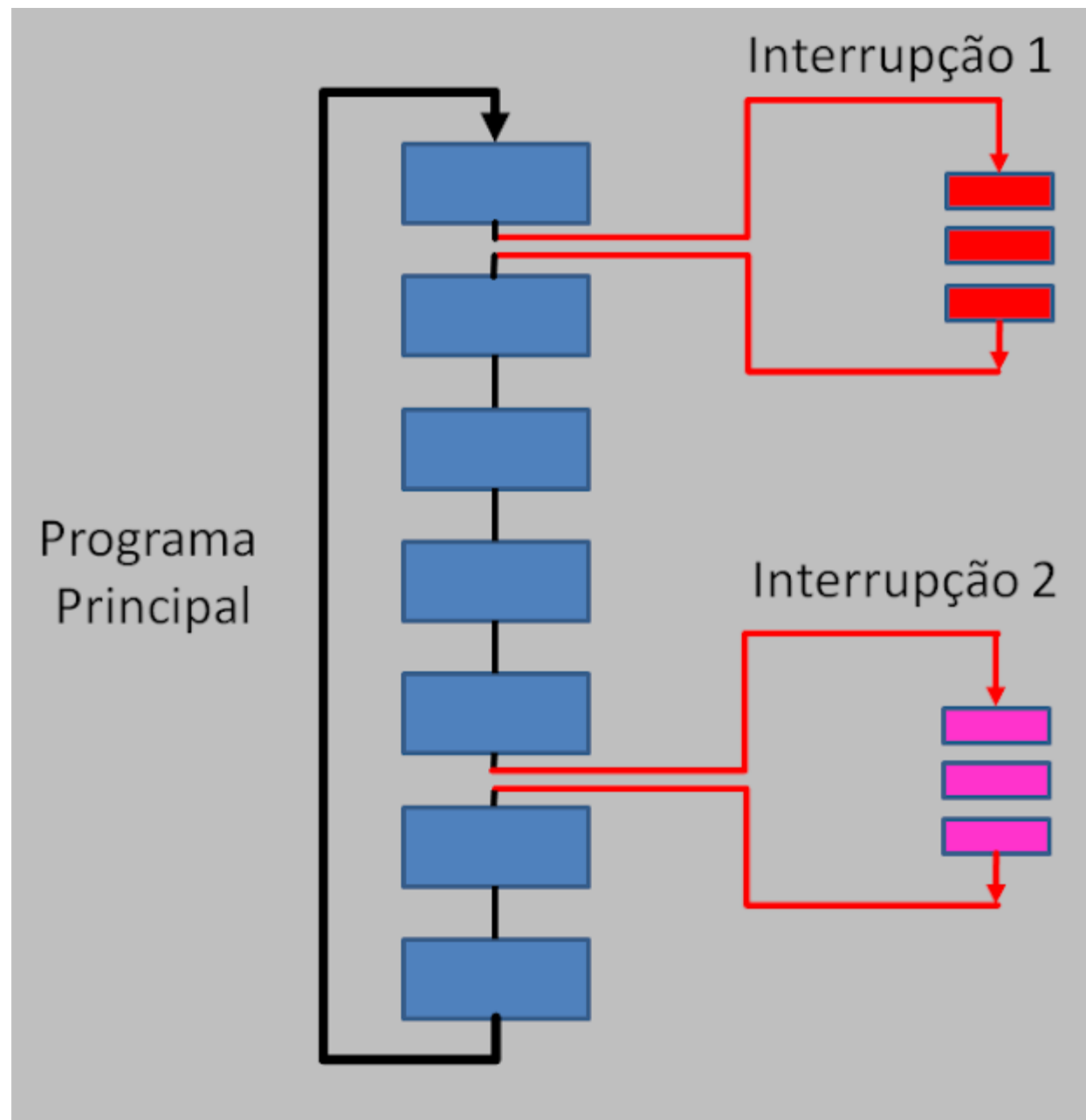
- Sem interrupções, o fluxo do programa é determinado pelo programa principal;
- **Problema:**
  - Alguns periféricos precisam de tempo para executar suas tarefas;
  - Espera ocupada: Não faz nada até que uma variável ou um flag mude de valor – pooling!





# INTERRUPÇÃO

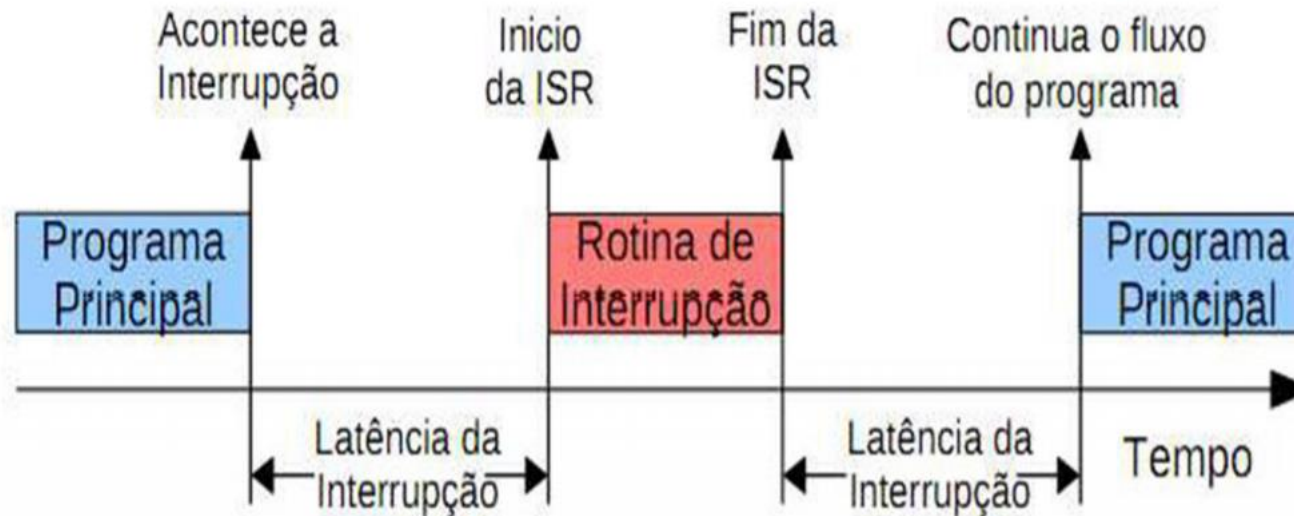
- **Solução:** Interrupção
- Sistema capaz de avisar quando uma determinada tarefa acabou.
- A tarefa então é executada, sem prejuízo para o fluxo do programa principal.



# INTERRUPÇÃO

- É um evento externo ao programa que provoca um desvio no seu fluxo, de forma que a CPU passa a executar um subprograma em resposta ao evento.
- Ao término do subprograma – ISR (Interrupt Servicing Routine) ou RTI (Rotina de Tratamento de Interrupção) – o fluxo do programa retorna ao ponto em que se encontrava antes da interrupção.
- Latência de interrupção: tempo decorrido entre o início do evento e o início da execução da RTI
- No MSP430 a latência de interrupção é fixa e determinada – são seis ciclos de clock para que a CPU reconheça a interrupção e efetue todo o procedimento de desvio do fluxo do programa.

# LATÊNCIA DE INTERRUPÇÃO

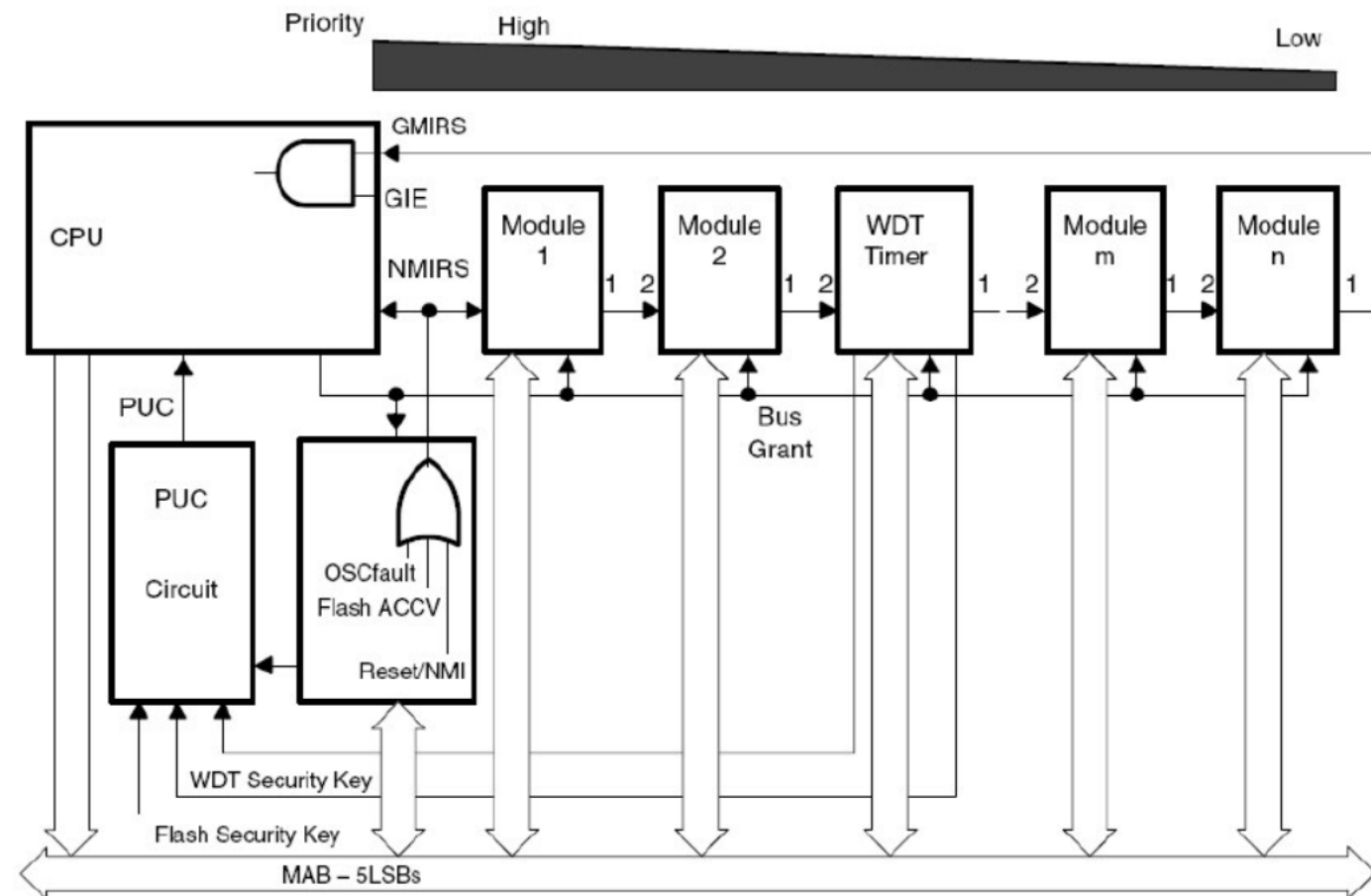


# CATEGORIAS DE INTERRUPÇÃO

- No MSP430 existem três tipos de interrupções:
  - RESET
  - NMI: interrupções não-mascaráveis
  - Interrupções mascaráveis

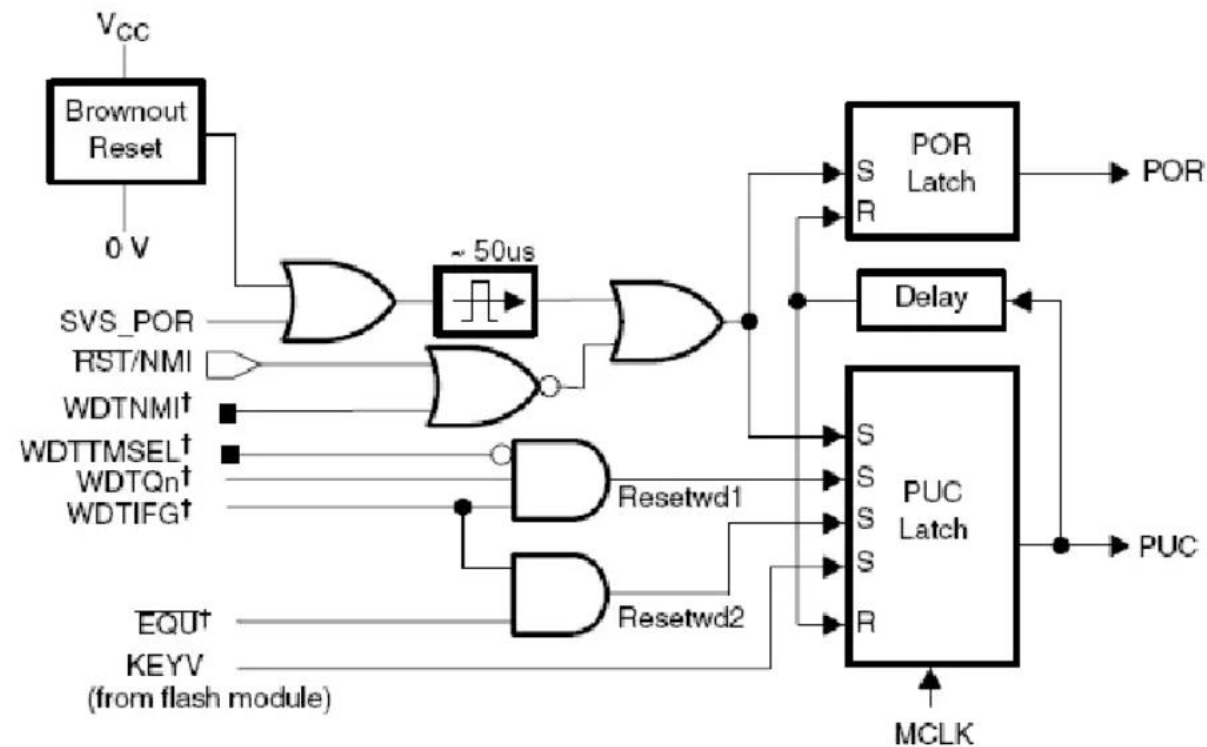


# CATEGORIAS DE INTERRUPÇÃO



# RESET

- É tratada como uma instrução não-mascarável.



† From watchdog timer peripheral module

# RESET

- POR (POWER ON RESET) – Reset na energização do chip: responsável pela inicialização de alguns registradores e condições do microcontrolador.
- Gerado em três condições:
  - O dispositivo é alimentado;
  - Nível lógico 0 aplicado no pino RST/NMI (quando ativo na função reset pelo programador);
  - Nível de tensão no módulo SVS (supervisão de alimentação) – quando ativo o módulo detecta uma queda de tensão de alimentação (quando ativado pelo programador).

# RESET

- PUC (POWER UP CLEAR) – confirmação de alimentação estável.
- Gerado pelos seguintes eventos:
  - POR ativo: um evento de power on reset dispara um sinal PUC;
  - Estouro na contagem do Watchdog Timer quando configurado para o modo reset;
  - Violação do watchdog – acesso indevido aos registradores de controle do WDT;
  - Violação de acesso aos registradores de controle do controlador da FLASH
  - BOR (Brown-Out Reset) – circuito que fica monitorando a tensão de alimentação do microcontrolador – fica monitorando se a tensão caiu por um dado valor por um determinado tempo.
- Watchdog Timer: garante o funcionamento do microcontrolador, resetando-o caso ele trave.

# RESET

- Efeitos do RESET:
  - O pino RST/NMI é configurado para o modo reset;
  - As portas de E/S são configuradas para a função de entrada digital;
  - O SR é apagado – todos os bits em nível lógico 0;
  - o WDT é ativado no modo reset;
  - O PC é carregado com o conteúdo indicado pelo vetor de reset – o programa volta para o começo.

# INTERRUPÇÕES NÃO-MASCARÁVEIS

- Não são controladas pelo usuário.
- Não podem ser desativadas pelo controle geral de interrupções – o bit GIE do SR.
- Três eventos capazes de gerar uma interrupção não-mascarável:
  - Nível 0 no pino RST/NMI – quando configurado no modo reset.
  - Falha no oscilador.
  - Violação de acesso a FLASH.



# INTERRUPÇÕES MASCARÁVEIS

- Dependem que o bit GIE esteja em nível lógico 1.
- Causadas por periféricos: Timers, USART, I2C, ESP430 etc

# PROCEDIMENTOS DA CPU

A instrução em execução é completada.



O conteúdo do PC é salvo na pilha – para que o programa possa retornar ao ponto em que se encontrava.



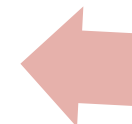
O conteúdo do SR é salvo na pilha – para preservar os bits de modo de operação e flags matemáticos da CPU.



Escolhe a interrupção de mais alta prioridade – em caso de múltiplas interrupções.



O flag da interrupção é apagado pela CPU.



O conteúdo de SR é apagado (exceto o bit SCG0) – isso faz com que o chip saia de um eventual modo de baixo consumo. O bit GIE é apagado – novas interrupções não-mascaráveis são impedidas de ocorrer.



O vetor correspondente à interrupção acionada é carregado no PC – provocando o desvio do fluxo do programa para a RTI correspondente.

# PROCEDIMENTOS DA CPU

## MSP430 - Interrupções

### Sequência de atendimento de uma interrupção

ENDEREÇO	MEMÓRIA FLASH
0xC000	Instrução 01
0xC002	Instrução 02
0xC004	Instrução 03
0xC006	Instrução 04
0xC010	Instrução 01
0xC012	Instrução 02
0xC014	RETI
0xFFFA	0xC010
0xFFFC	0XXXX
0xFFFE	0xC000

ENDEREÇO	MEMÓRIA RAM
0x0200	
0x0202	
0x0204	
0x0206	
0x0208	

Registradores da CPU	
PC	
SP	
SR	

Programa Principal

Rotina de interrupção

Vetor de Interrupção

Memória RAM

Área vazia

Registradores da CPU

# FINALIZAÇÃO DA INTERRUPÇÃO

1. A rotina de interrupção é finalizada com a instrução RETI;
2. O conteúdo do registrador de *status* (SR) é recuperado da pilha, retomando a sua configuração inicial (bit GIE, modo de operação, etc);
3. O conteúdo do contador de programa (PC) é recuperado da pilha e o programa principal volta a ser executado de onde ele havia parado.

# VETOR DE INTERRUPÇÃO

- Cada evento capaz de gerar uma interrupção deve ter a sua própria função para tratar desta interrupção;
- Para que a CPU possa saber onde está cada função de interrupção, existe uma tabela para guardar o endereço da função de cada interrupção;
- Esta tabela é chamada de Vetor de Interrupção.

# VETOR DE INTERRUPÇÃO

- Está localizado na região de memória FLASH entre 0xFFC0 e 0xFFFF;
- Contém o endereço da rotina de cada fonte de interrupção;
- O conteúdo de cada posição deve ser programado pelo desenvolvedor (compilador) com o endereço da rotina de interrupção correspondente;
- Cada fonte de interrupção possui sua posição específica neste vetor;
- A posição neste vetor determina a prioridade da interrupção. Quanto mais alta a posição, maior será a prioridade.



# VETOR DE INTERRUPÇÃO

INTERRUPT SOURCE	INTERRUPT FLAG	SYSTEM INTERRUPT	WORD ADDRESS	PRIORITY
Power-up External reset Watchdog Timer+ Flash key violation PC out-of-range (see Note 1)	PORIFG RSTIFG WDTIFG KEYV (see Note 2)	Reset	0FFFEh	31, highest
NMI Oscillator fault Flash memory access violation	NMIIFG OFIFG ACGVIFG (see Notes 2 and 4)	(non)-maskable, (non)-maskable, (non)-maskable	0FFFCh	30
			0FFFAh	29
			0FFF8h	28
Comparator_A+ (MSP430x20x1 only)	CAIFG (see Note 3)	maskable	0FFF6h	27
Watchdog Timer+	WDTIFG	maskable	0FFF4h	26
Timer_A2	TACCR0 CCIFG (see Note 3)	maskable	0FFF2h	25
Timer_A2	TACCR1 CCIFG. TAIFG (see Notes 2 and 3)	maskable	0FFF0h	24
			0FFEEh	23
			0FFEC	22
ADC10 (MSP430x20x2 only)	ADC10IFG (see Note 3)	maskable	0FFEAh	21
SD16_A (MSP430x20x3 only)	SD16CCTL0 SD16OVIFG, SD16CCTL0 SD16IFG (see Notes 2 and 3)	maskable		
USI (MSP430x20x2, MSP430x20x3 only)	USIIFG, USISTTIFG (see Notes 2 and 3)	maskable	0FFE8h	20
I/O Port P2 (two flags)	P2IFG.6 to P2IFG.7 (see Notes 2 and 3)	maskable	0FFE6h	19
I/O Port P1 (eight flags)	P1IFG.0 to P1IFG.7 (see Notes 2 and 3)	maskable	0FFE4h	18
			0FFE2h	17
			0FFE0h	16
(see Note 5)			0FFDEh ... 0FFC0h	15 ... 0, lowest

# EXEMPLO — PISCA LED

```
1  #include <msp430x16x.h>
2
3  ;Objetivo: piscar o LED conectado ao pino P1.0 configurando o TIMER A
4  ;          gerar uma interrupção periódica
5
6      NAME main
7      PUBLIC main
8      ORG 0xFFEA          ;vetor da interrupção TAIFG
9      DC16 trata_timer    ;define o endereço da RTI
10     ORG 0xFFFFh
11     DC16 main
12     RSEG CODE
13
14 main MOV #0x3FE,SP      ;inicializa o apontador da pilha
15     MOV #WDTPW+WDTHOLD,WDCTL ;desliga o watchdog
16     MOV #TASSEL_1+ID_3+MC_1+TAIE,TACTL ;configura o timer
17     MOV #0x8000,TACCR0   ;configura o módulo de contagem
18     MOV.B #1,P1DIR       ; configura o P1.0 como saída
19     EINT                 ;habilita interrupções
20 fim JMP fim             ;loop infinito
21
22 trata_timer              ;RTI
23     XOR.B #1,P1OUT        ;inverte o estado do pino P1.0
24     RETI
25     END main
```

# ATIVIDADE PRÁTICA

- Implementar o exemplo anterior (Pisca LED) fazendo com que tenha salvamento de contexto na entrada da RTI e restauração do contexto na saída.
- A função principal deve ter alguma aplicação (o código do livro de Fábio tem um contador baseado no registrador R9).
- A sequência de salvamento dos registradores deve ser respeitada no instante da restauração dos valores, ou seja, o último registrador salvo deve ser o primeiro a ser recuperado e assim por diante, até que o primeiro registrador salvo seja recuperado.
- O circuito pode ser simulado no Proteus.
- Prazo - Esquema, código e apresentação: **12/04/2023**
- **Um membro da equipe será escolhido para realizar a apresentação e os demais deverão responder a questionamentos após a apresentação.**
- Tempo de apresentação: 10 minutos