

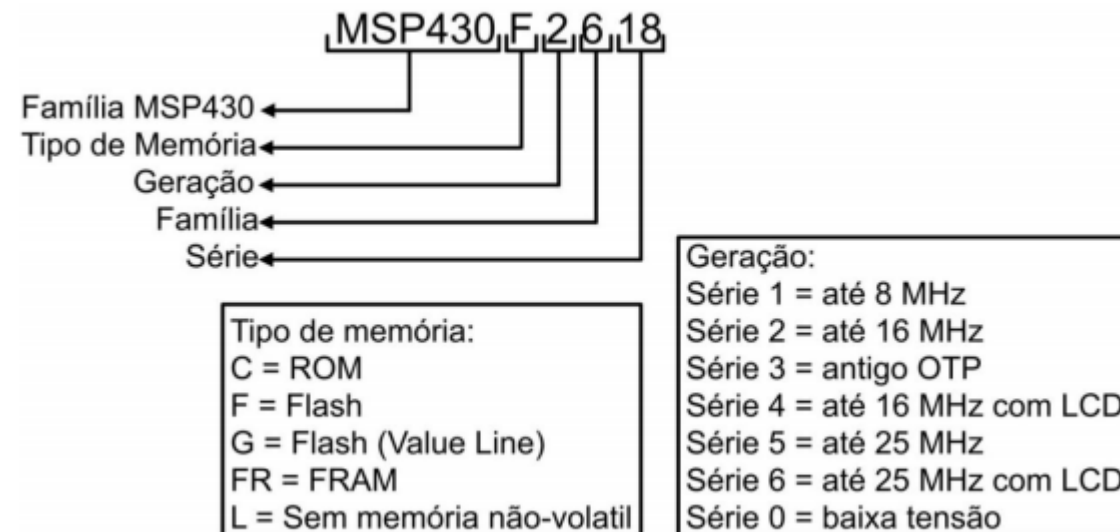
# MSP430 — KIT DE DESENVOLVIMENTO LAUNCH PAD



# MSP430

- A família MSP430 foi desenvolvida pela Texas Instruments na década de 1990.
- Microcontroladores de baixa potência.
- Utiliza arquitetura **Von Neumann**.
- Conjunto de instruções **RISC**.
- Conjunto de instruções formado por 27 instruções físicas mais 24 emuladas = 51 instruções.

# MSP430 - NOMENCLATURA



# MSP430 - CPU

- Barramentos:
  - Endereços: 16 bits – pode acessar até 65536 posições de memória.
  - Dados: 16 bits – pode processar informações em lotes de 16 bits
  - Controle

# MSP430 - CPU

- A CPU possui 16 registradores de 16 bits:
  - Um contador de programa (R0) – PC – Program Counter
  - Um ponteiro de pilha (R1) – SP – Stack Pointer
  - Um registrador de status que pode ser usado como gerador de constantes (R2) – SR – Status Register ou CG1 – Constant Generator
  - Um gerador de constantes (R3) – CG2
  - Doze registradores de propósito geral (R4 a R15) - GPR – General Purpose Registers

# CONTADOR DE PROGRAMA (R0/PC)

- Possui a finalidade de apontar a próxima instrução a ser lida da memória e executada pela CPU.
- O espaço total de endereçamento é de 64K ou 65536 endereços.
- O espaço de endereçamento é organizado em bytes.
- As instruções ficam localizadas sempre nos endereços pares da memória.

# APONTADOR DE PILHA (R1/SP)

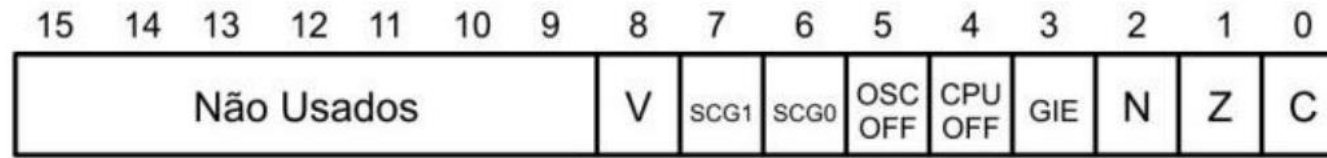
- Usado para indicar à CPU a localização do topo da pilha de memória.
- A pilha de memória, ou simplesmente pilha, é utilizada para o armazenamento de endereços de retorno nas chamadas de sub-rotinas e tratamento de interrupções.

# 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.

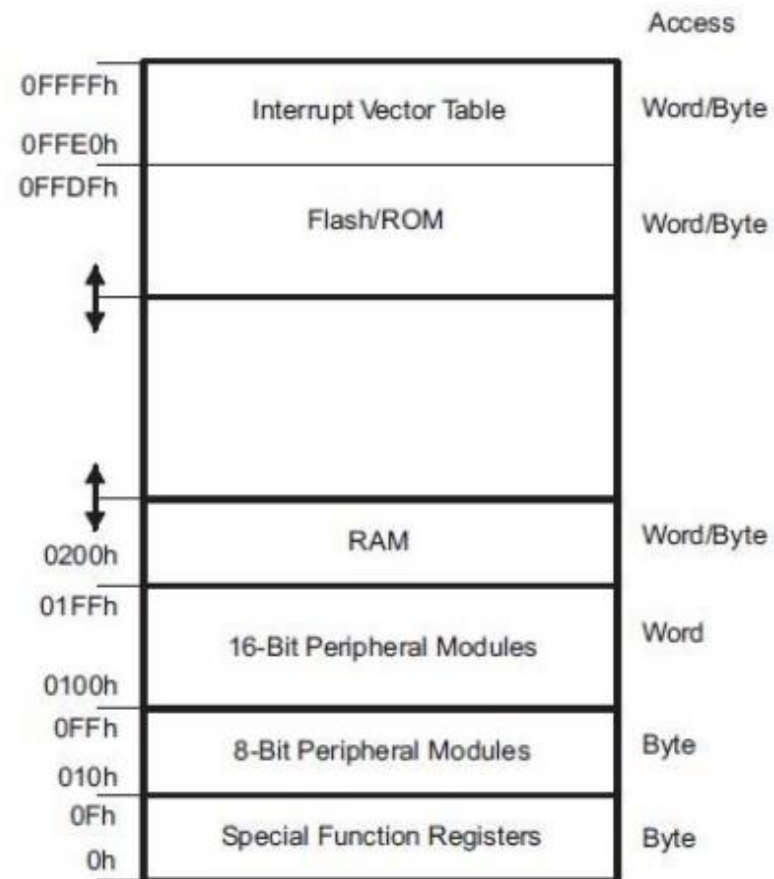
# REGISTRADORES GERADORES DE CONSTANTES (R2 E R3)

- Responsáveis pela geração de constantes numéricas necessárias à emulação de instruções, que consiste numa extensão do conjunto físico de instruções.

# REGISTRADORES DE PROPÓSITO GERAL (R4 A R15)

- Podem ser utilizados para funções diversas à escolha do usuário:
  - armazenamento de variáveis de uso intensivo;
  - apontadores de endereço (ponteiros);
  - etc.

# MSP430 — ORGANIZAÇÃO DE MEMÓRIA



# MSP430 — ORGANIZAÇÃO DE MEMÓRIA

- Registradores de funções especiais: Habilitar funções em alguns módulos.
- Registradores periféricos: responsáveis por fazer a comunicação entre CPU e periféricos.
- RAM: memória de dados utilizada para armazenar as variáveis do programa escritas pelo usuário. O tamanho depende do dispositivo.
- ROM: armazena o programa escrito pelo usuário. O início da faixa e quantidade de memória dependem do dispositivo.
- Tabela do vetor de interrupções: região da memória ROM que armazena os endereços das rotinas de tratamento de interrupções.

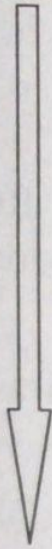
# MSP430 - PERIFÉRICOS

- Possuem uma gama muito ampla de periféricos:
  - conversores AD/DA;
  - amplificador operacional programável;
  - timers;
  - Watchdog timer
  - entre outros.

# MSP430 — MODOS DE OPERAÇÃO

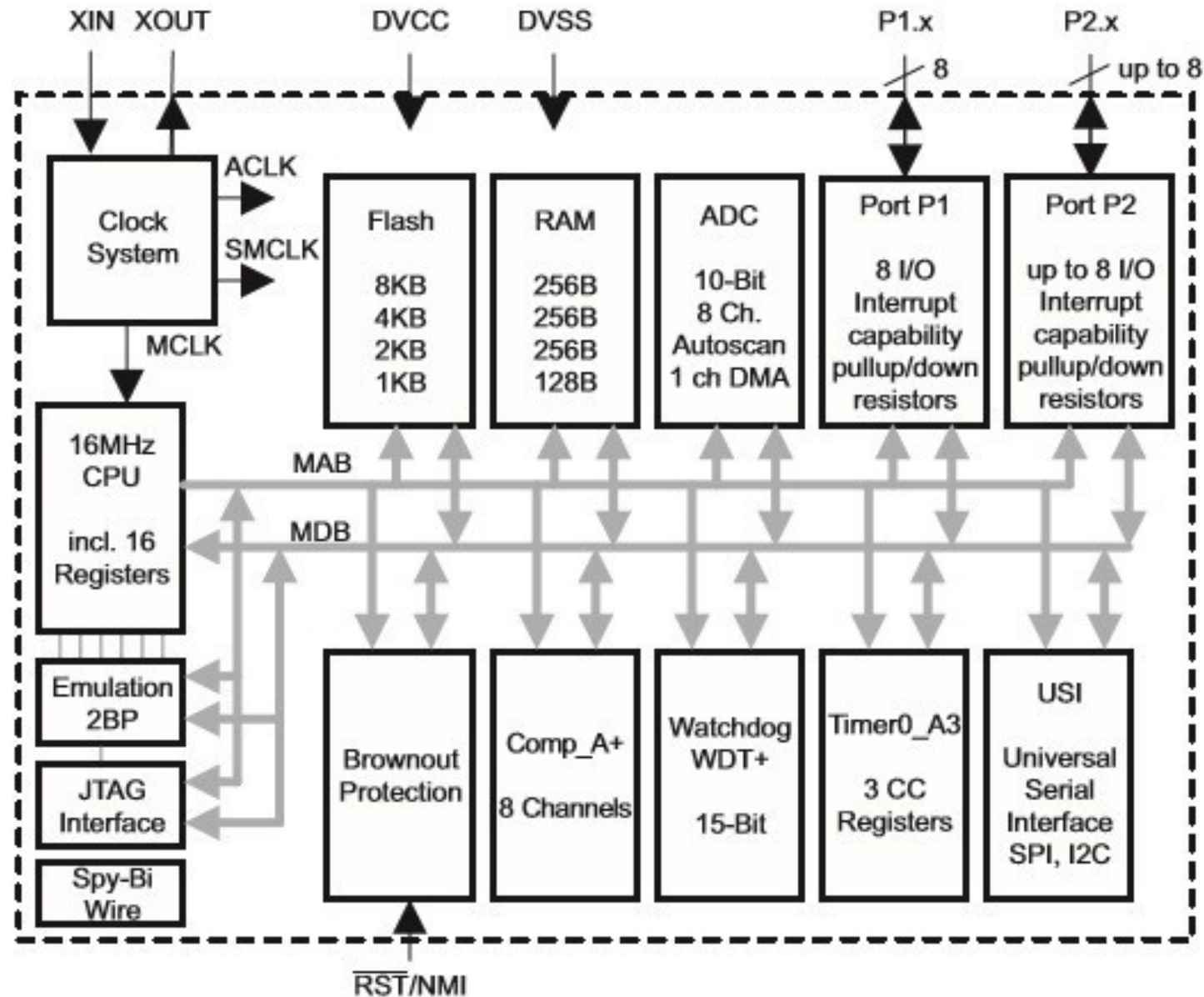
- São disponibilizados modos de funcionamento que permitem um controle bastante preciso do consumo de corrente pelo chip.
- São 6 modos de operação controlados de acordo com o estado dos bits CPUOFF, OSCOFF, SCG0 e SCG1 do R2.

# 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



## Functional Block Diagram, MSP430G2x52



# CONJUNTO DE INSTRUÇÕES

- Instruções com um operando;
- Instruções com dois operandos;
- Instruções de desvio (salto).

# ASSEMBLY

- Assembly é uma linguagem de baixo nível, chamada frequentemente de “linguagem de montagem”
- É uma linguagem considerada difícil, principalmente porque o programador precisa conhecer a estrutura da máquina para usá-la

# ASSEMBLY

- A linguagem Assembly é atrelada à arquitetura de uma certa CPU, ou seja, ela depende completamente do hardware
- Cada família de processador tem sua própria linguagem assembly (Ex. X86, ARM, SPARC, MIPS)
- Por essa razão Assembly não é uma linguagem portátil, ao contrário da maioria das linguagens de alto nível

# ASSEMBLY

- As primeiras linguagens Assembly surgiram na década de 50, na chamada segunda geração das linguagens de programação
- A segunda geração visou libertar os programadores de dificuldades como lembrar códigos numéricos e calcular endereços

# ASSEMBLY

- Assembly foi muito usada para várias aplicações até os anos 80, quando foi substituída pelas linguagens de alto nível
- Isso aconteceu principalmente pela necessidade de aumento da produtividade de software
- Atualmente Assembly é usada para manipulação direta de hardware e para sistemas que necessitem de performance crítica
- Device drivers, sistemas embarcados de baixo nível e sistemas de tempo real são exemplos de aplicações que usam Assembly

# ASSEMBLY

- A linguagem Assembly é de baixo nível, porém ainda precisa ser transformada na linguagem que a máquina entende
- Quem faz isso é o Assembler. O Assembler é um utilitário que traduz o código Assembly para a máquina

# ASSEMBLY

- Exemplo:

mov -> copia os dados da fonte para o destino

Antes -> mov al, 061h (x86/IA-32)

Depois -> 10110000 01100001



# APRESENTAÇÃO CONJUNTO DE INSTRUÇÕES MSP430

- Temas.
  1. Instruções com um operando –;
  2. Instruções com um operando
  3. Instruções com dois operandos –;
  4. Instruções com dois operandos
  5. Instruções de desvio (salto) –;
  6. Instruções Emuladas –.
  7. Instruções Emuladas
- Deve apresentar exemplos de instruções no IAR.
- As equipes com o mesmo tema devem dividir as instruções entre si.
- A participação de todos os integrantes é obrigatória.
- As apresentações ocorrerão nos dias 13/03 (Equipes 1,2,3 e 4) e 15/03 (Equipes 5,6 e 7)
- Pontuação: 2 pontos na AV1

# INSTALAR IAR

- [https://www.iar.com/products/architectures/iar-embedded-workbench-for-msp430/#containerblock\\_3096](https://www.iar.com/products/architectures/iar-embedded-workbench-for-msp430/#containerblock_3096)



# REFERÊNCIA

- **Ciro Ceissler. Notas de aula**