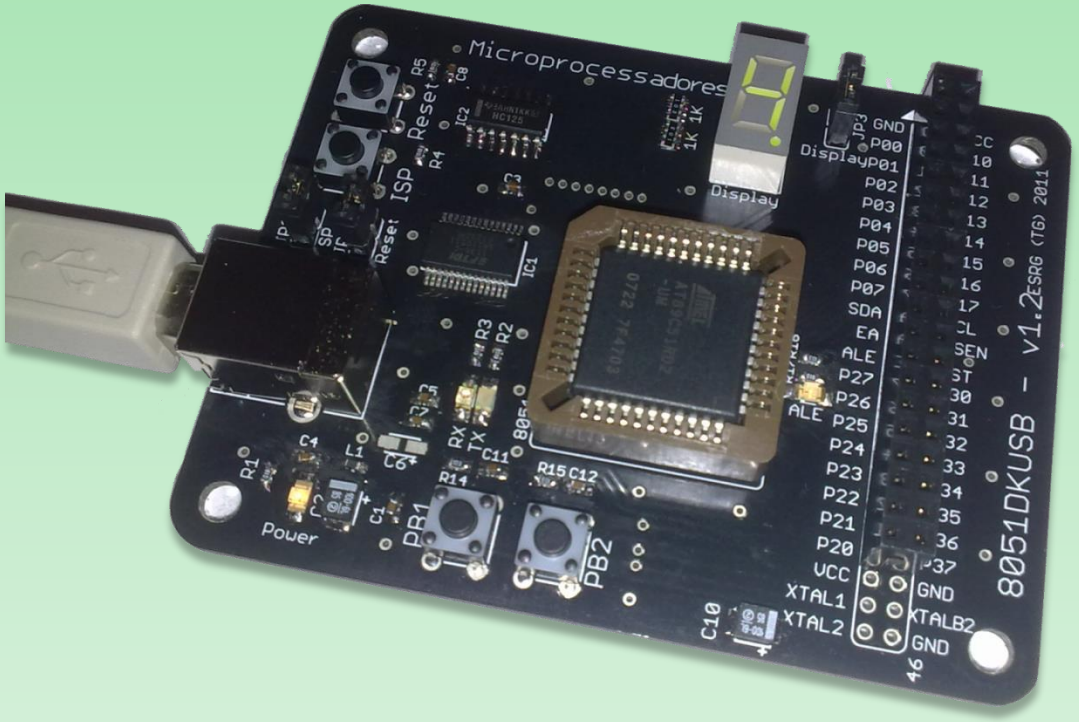
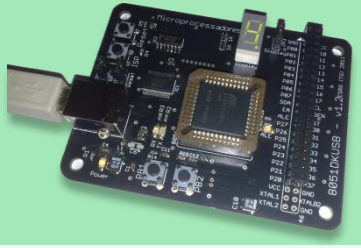
Mestrado Integrado em Eng. Engenharia de Telecomunicações e Informática



Interrupções

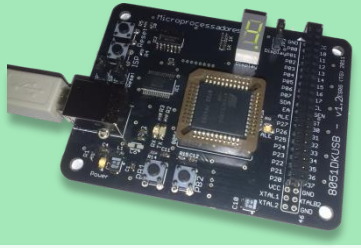
Microcontroladores

2º Ano – A11



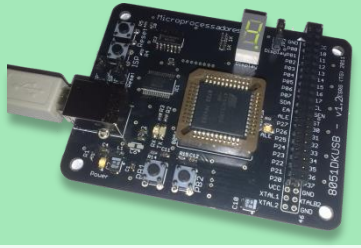
Interrupções: *O que são?*

- Interrupção
  - Qualquer evento que interrompe a execução normal do CPU:
    - Externo ou interno ao processador
      - Exemplos:Alteração do valor num pino externo do microcontrolador;
        - » Recepção de dados porto série;
        - » *Overflow* de um temporizador.
  - Podemos utilizar o *polling* para lidar com os eventos:
    - i.e. ficar à espera que o evento ocorra (em teste contínuo)



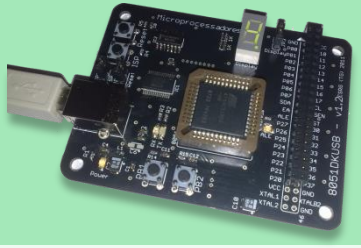
Interrupções

- Interrupções
  - Externas:
    - Através da activação de sinais em pinos I/O do microcontrolador.
  - Internas:
    - *Traps* (não existem na família MCS-51)
      - Interrupção provocada pelo programador, usando uma instrução do ISA.
    - Excepções (não existem na família MCS-51)
      - Interrupções provocadas pela execução do código. Exemplo: divisão por zero.
    - Periféricos internos (microcontroladores)
      - Um dos periféricos internos, envia um sinal ao CPU, exemplos:
        - » *Overflow* de uma unidade contadora/temporizadora;
        - » Final de transmissão série.



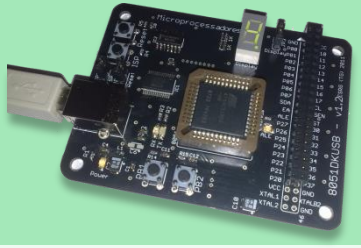
Interrupções

- Interrupção - caracterização
  - Sinal a pedir a atenção do processador;
  - Tratamento do sinal por parte do CPU;
  - Atendimento da interrupção:
    - Por software
      - Rotinas de código, semelhantes às sub-rotinas, que são invocadas automaticamente pelo CPU em resposta ao sinal da interrupção. São designadas por ISR (*Interrupt Service Routine*).
  - Quando surge um sinal de interrupção o CPU não interrompe a instrução actual:
    - O sinal é detectado pelo CPU durante o ciclo de execução (*fetch-decode-execute)* e, após o *execute*, o CPU pode ignorar ou atender a interrupção (programável, i.e., o programador decide).



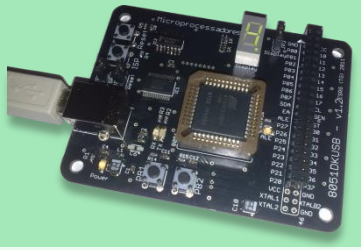
Interrupções

- Interrupção vectorizada:
  - Consoante a fonte do sinal, o CPU salta para uma posição da memória de código específica;
  - Vector de interrupção:
    - Definidos por hardware;
    - Espaço reservado é pequeno, pelo que apenas se coloca um salto para outra posição da memória de código.
- Interrupção não vectorizada:
  - Existe apenas um vector, o CPU salta sempre para a mesma posição de memória:
    - O programador testa as *flags* das interrupções para determinar qual o sinal que originou a interrupção.

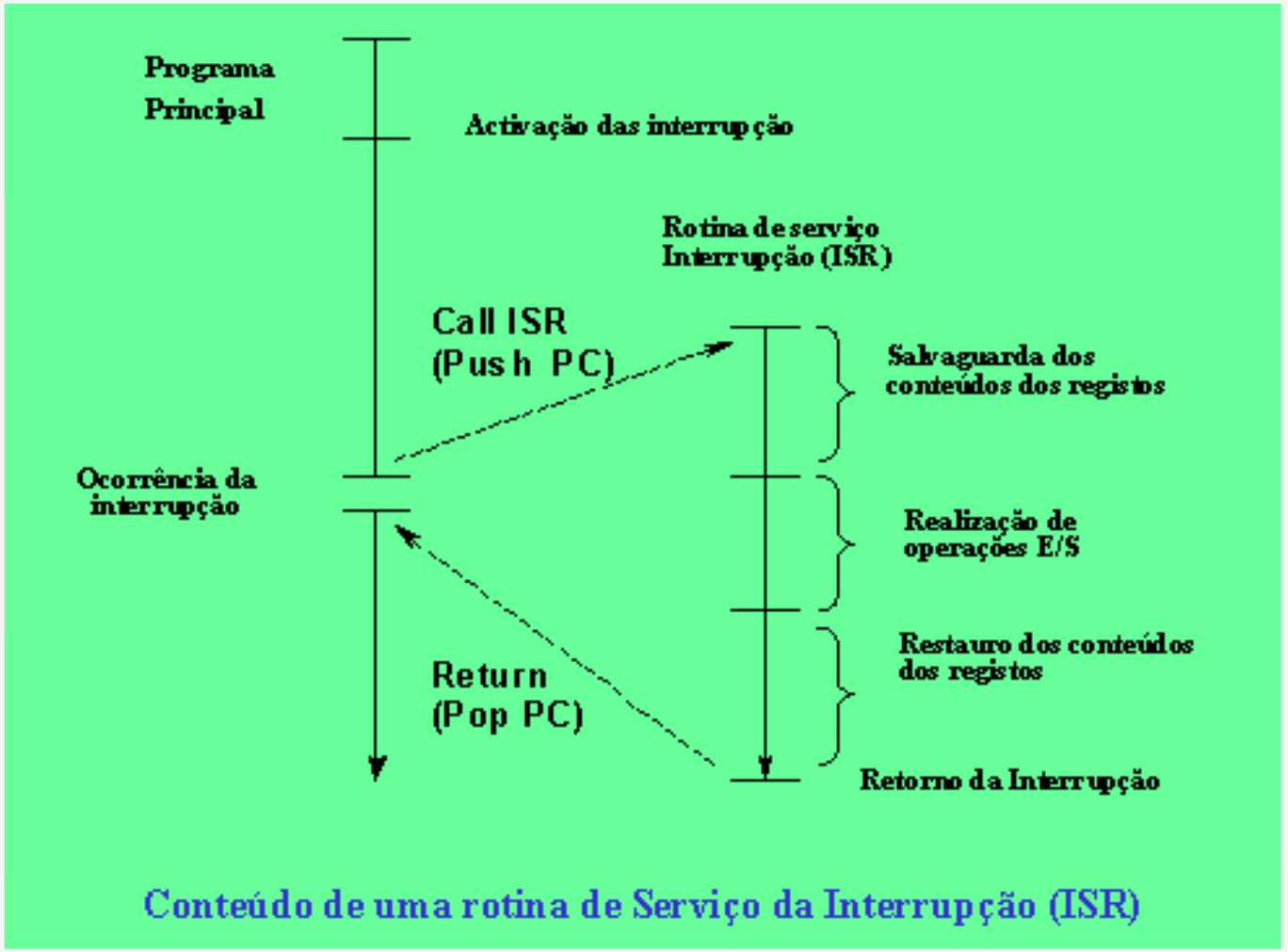


Interrupções: *O que são?*

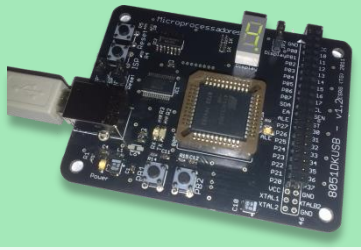
- As interrupções provocam uma transferência de execução para um endereço conhecido onde reside uma rotina de serviço à interrupção (ISR)



Interrupções: *O que são?*



Conteúdo de uma rotina de Serviço da Interrupção (ISR)



Interrupções: *O que são?*

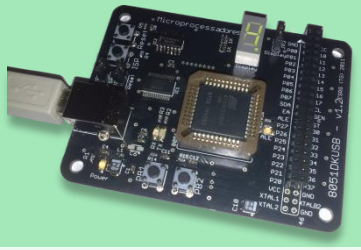
- Interrupção *versus Polling*

Vantagens do Polling

- *Simples de implementar;*
- *A execução está sincronizada com a execução do programa, o programador sabe exactamente quando é verificado o estado do dispositivo e quanto tempo será necessário para servi-lo;*
- *Não será necessário um sistema automático para salvaguardar o estado do processador;*
- *Não requer atenção à gestão da pilha.*

Vantagens da Interrupção

- *Não desperdiça tempo de execução na verificação contínua do estado dos dispositivos;*
- *O atendimento aos dispositivos pode ser imediato, fornecendo por isso capacidade de resposta em tempo-real.*



Interrupções: *O que são?*

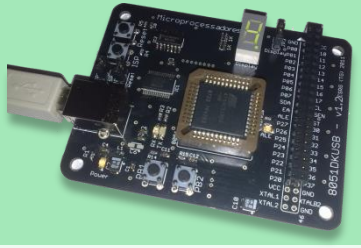
- Interrupção *versus Polling*

Desvantagens do Polling

- *Desperdiça tempo de execução, reduzindo por isso tempo disponível para outras actividades;*
- *Os dispositivos não podem ser atendidos de forma instantânea, tornando-se às vezes necessário mais recursos de armazenamento;*
- *Dispositivos menos prioritários podem nunca ser atendidos.*

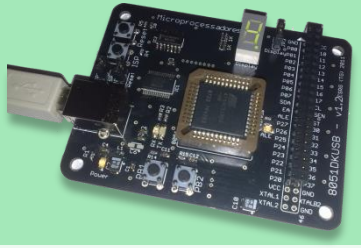
Desvantagens da Interrupção

- *Requer h/w extra;*
- *Introduz um overhead sempre que um dispositivo interrompe a execução do programa;*
- *A execução da ISR é assíncrona relativamente à execução do programa, dificultando a estimação da temporização da interrupção;*
- *A gestão de um número excessivo de interrupções pode tornar o sistema lento.*



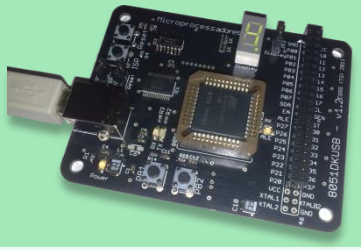
Interrupções: *O que são?*

- As interrupções podem ser activadas ou desactivadas.
  - O CPU recebe informação para ignorar as interrupções.
  - A maioria dos processadores possui uma instrução que informa o CPU para ignorar todas as interrupções “mascaráveis”.
    - Designado por desactivação global das interrupções.
  - O controlador de interrupções normalmente fornece facilidades para desactivar interrupções individualmente.



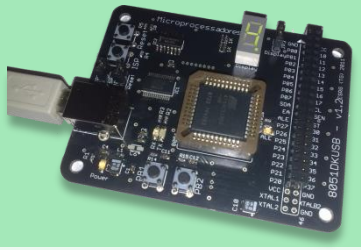
Interrupções: *O que são?*

- Quando se deve desactivar as interrupções?
  - Quando estamos a executar uma actividade que não deve ser interrompida, ou seja, quando a actividade deve ser considerada como atómica.



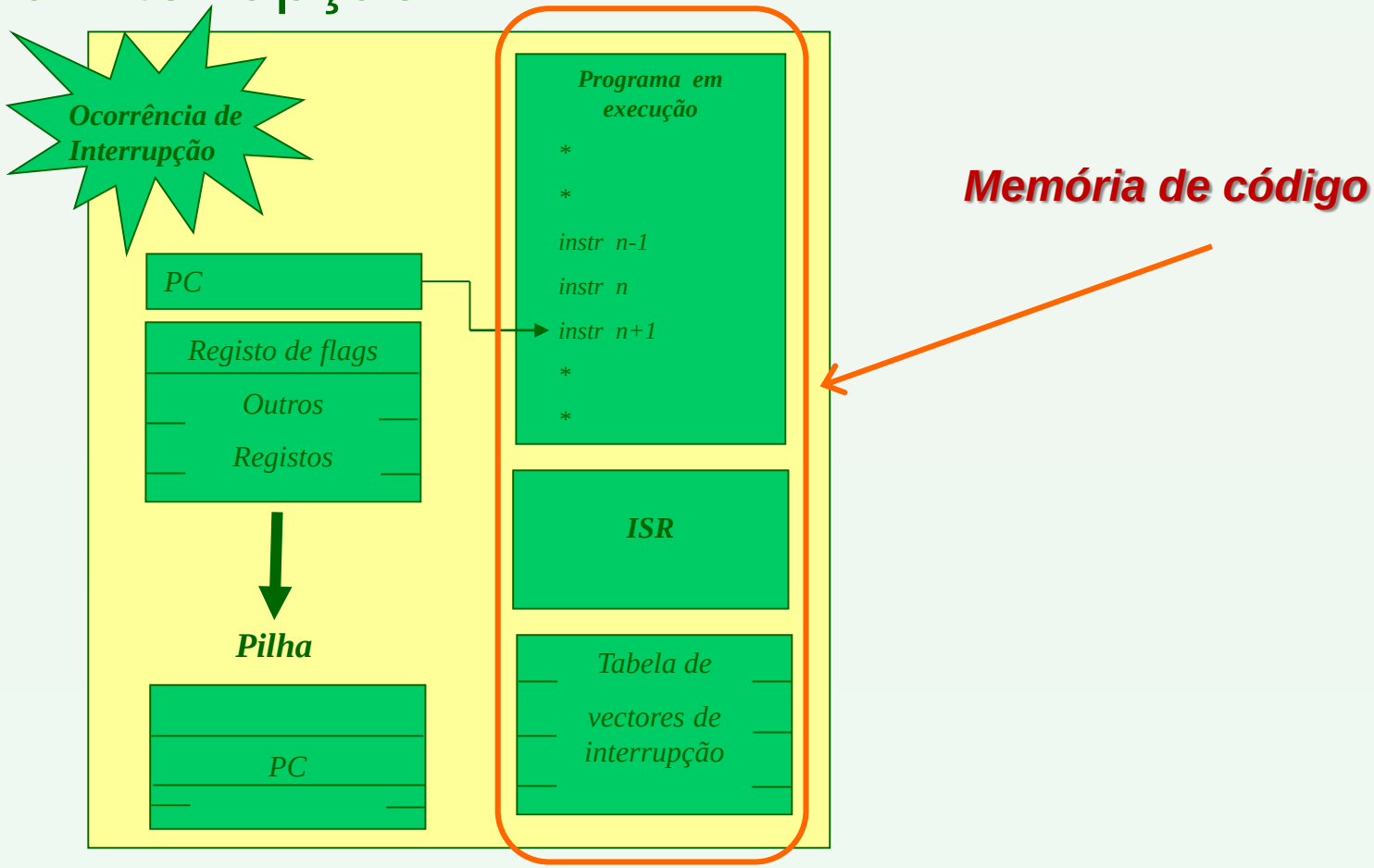
Interrupções: *O que são?*

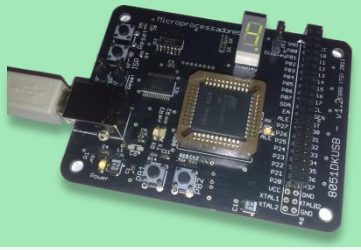
- Como é que o CPU sabe qual a ISR a executar aquando da ocorrência de uma interrupção?
  - Cada interrupção tem associado um número de identificação.
  - O CPU usa este número de interrupção para consultar a tabela de vectores de interrupção e obter o endereço da ISR correcta.
    - A tabela de vectores é uma colecção de apontadores para as ISRs.
    - O número da interrupção serve de índice na tabela de vectores de interrupção.
    - Cada entrada da tabela de vectores de interrupção tem um número fixo de *bytes* que é suficientemente grande para armazenar um endereço de memória.



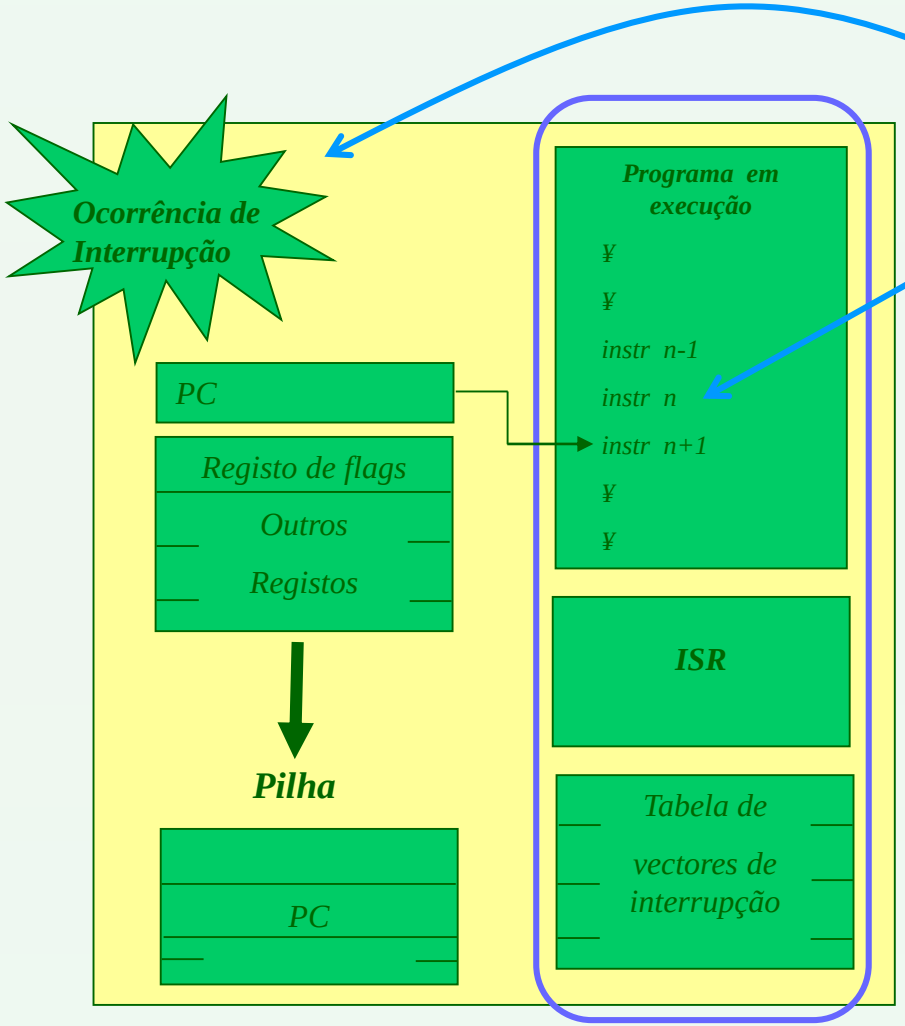
Interrupções: *O que são?*

- O que acontece logo no início da ocorrência de uma interrupção?

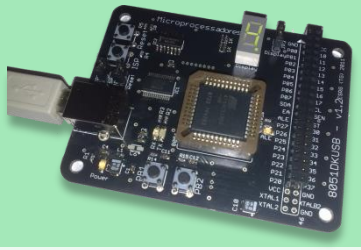




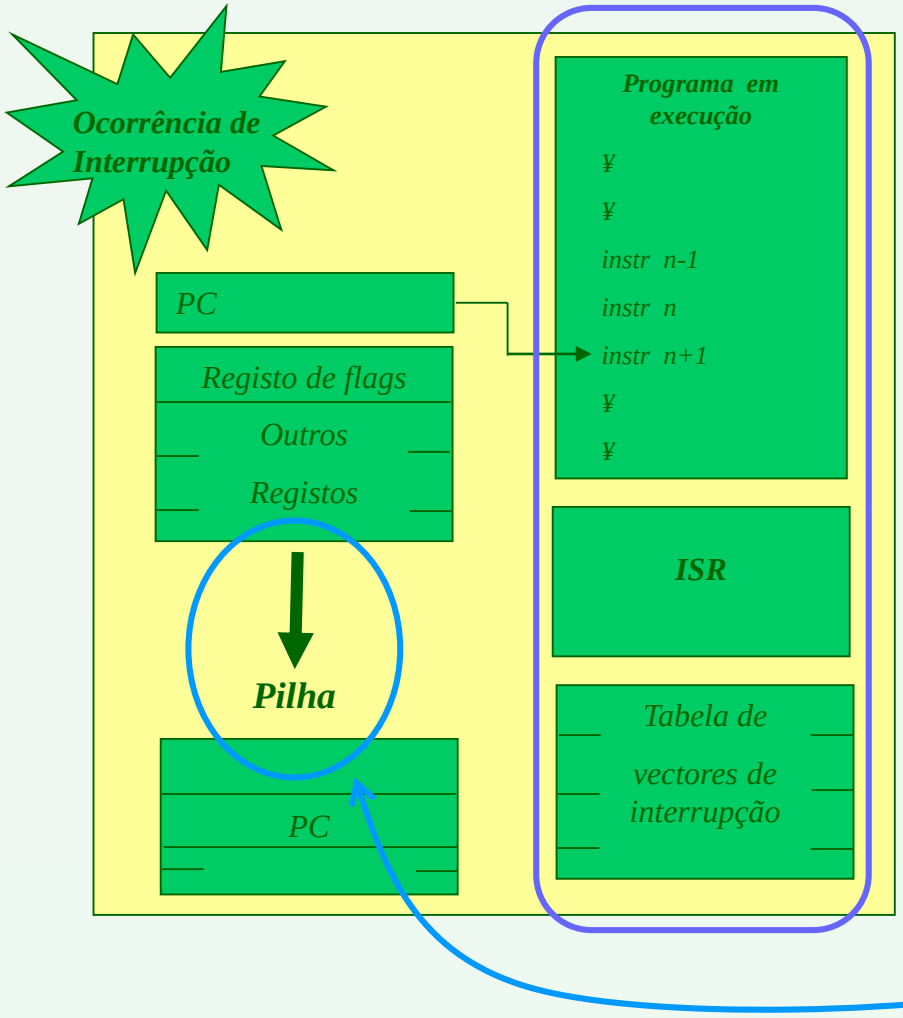
Interrupções: *O que são?*



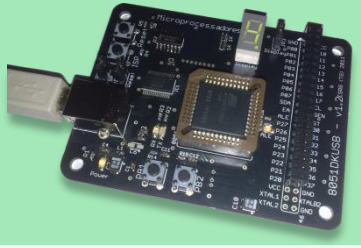
- *Uma interrupção é desencadeada durante a execução da instrução n.*



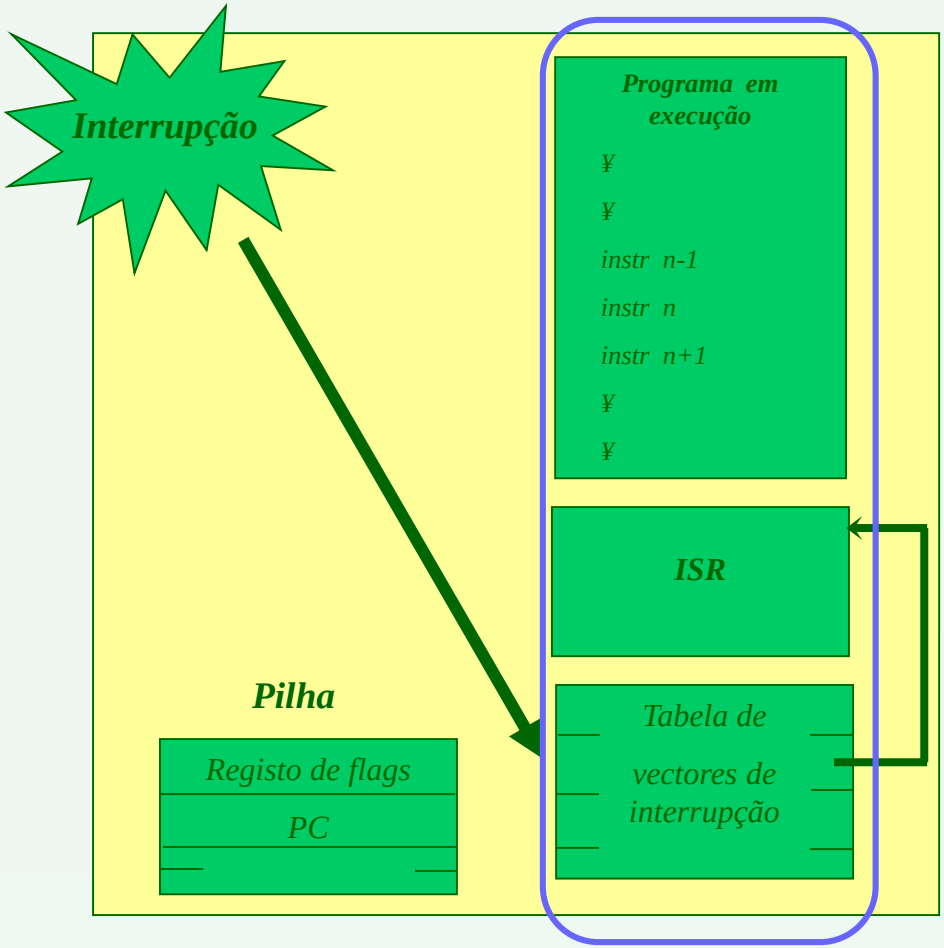
Interrupções: *O que são?*



- *O sistema conclui a execução da instrução n.*
- *Como resposta, o sistema salvaguarda o PC actual na pilha (alguns processadores armazenam também o estado do CPU)*

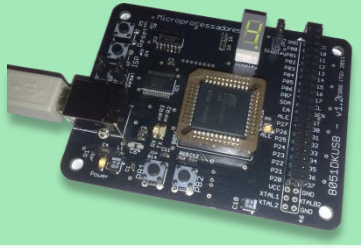


Interrupções: *O que são?*

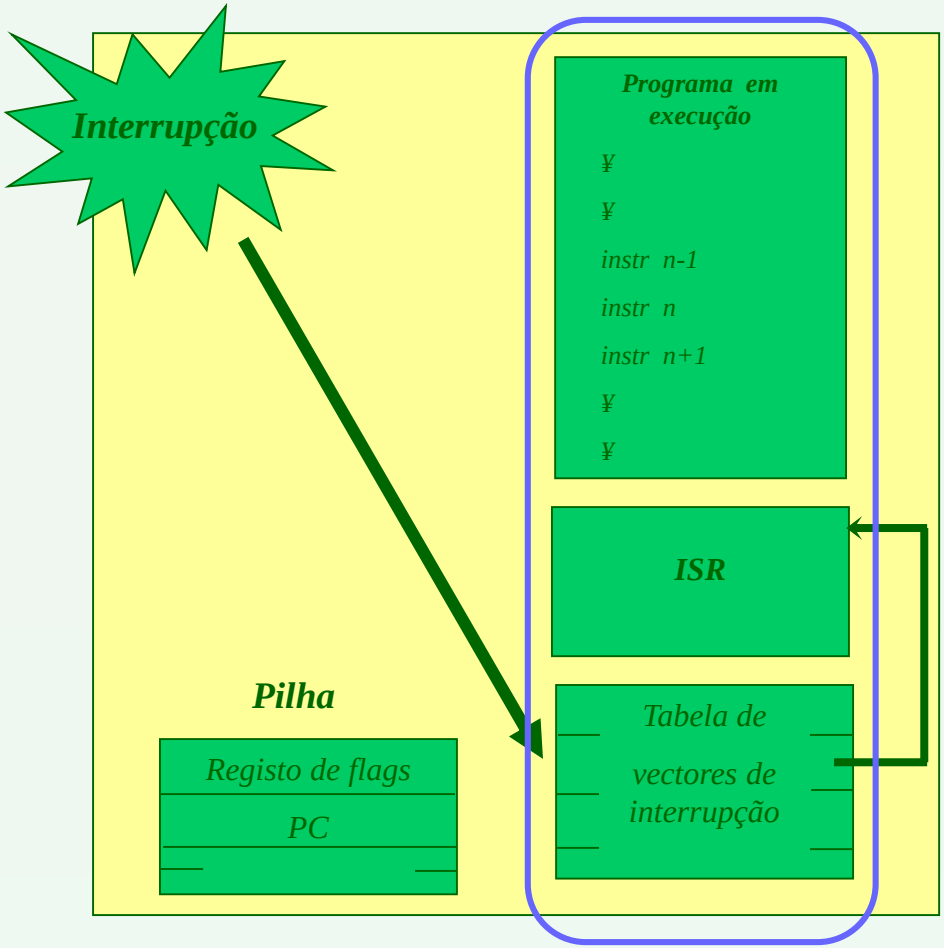


A tabela de vectores é uma colecção de apontadores para as ISRs.

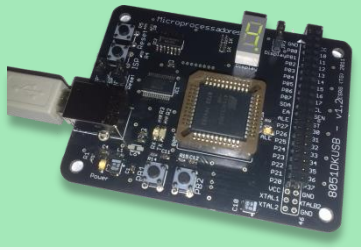
- *O número da interrupção serve de índice na tabela de vectores de interrupção*
  - *Cada apontador corresponde a um número de interrupção.*



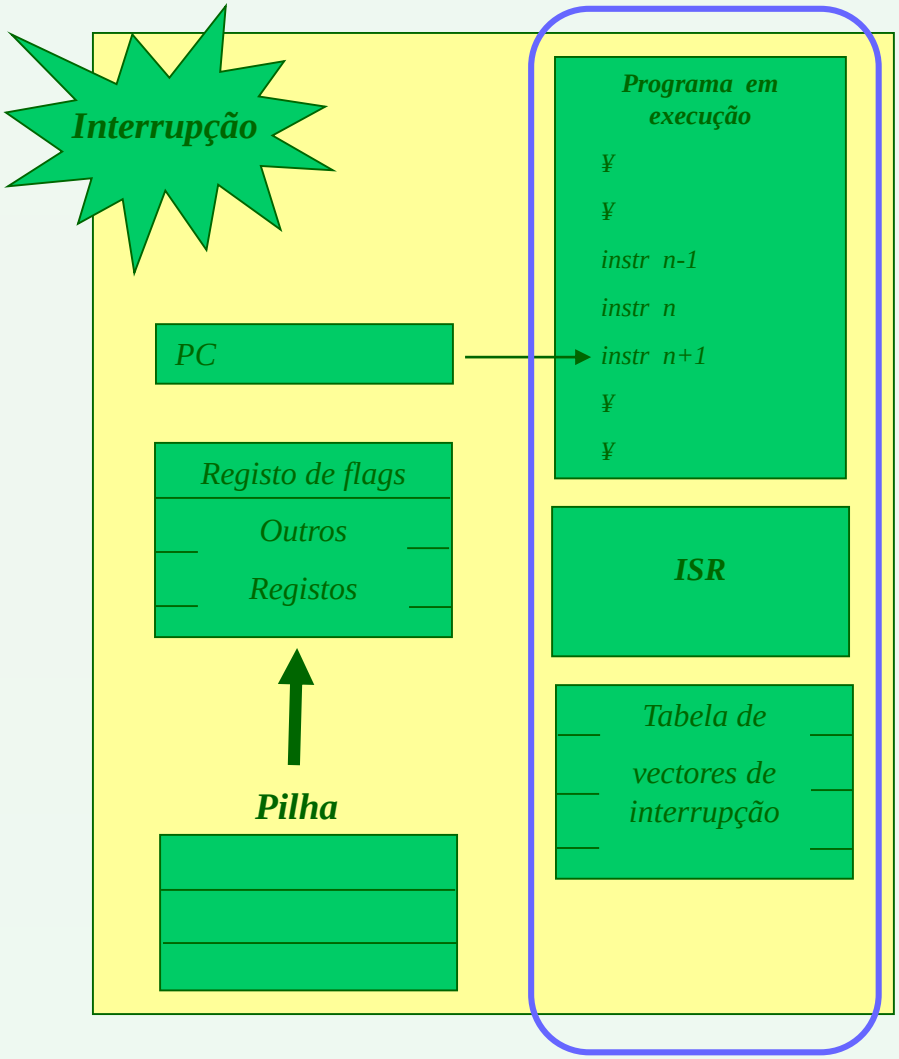
Interrupções: *O que são?*



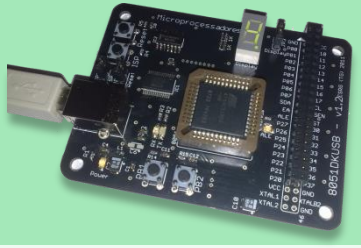
- *Alguns processadores usam uma tabela de saltos em vez da tabela de vectores de interrupção.*
  - *Cada entrada na tabela deve ser capaz de armazenar uma instrução de salto.*
  - *O processador salta para uma entrada da tabela e literalmente será redireccionado para a ISR dado que a entrada é uma instrução de salto.*



Interrupções: *O que são?*

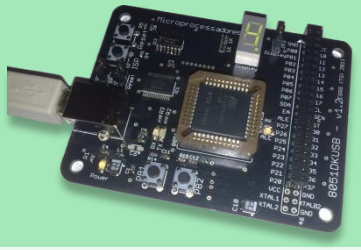


- *Ao concluir a execução, a ISR restaura da pilha os registos que o programa interrompido usava*
- *Restaura também o registo PC*
  - *Esta última operação devolve o controlo ao programa interrompido*



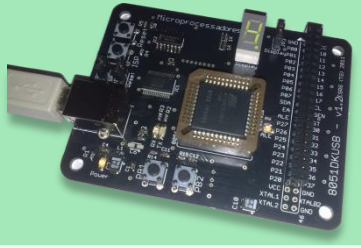
Interrupções: *O que são?*

- Quando se deve desactivar determinada interrupção em vez de desactivar todas as interrupções ?
  - Deve-se mascarar uma interrupção caso não seja usada, evitando deste modo a possibilidade de ocorrência de falsas interrupções.
    - A maioria das aplicações desactivam todas as interrupções que não estejam em uso durante a inicialização.
  - Uma ISR deve mascarar a sua própria interrupção para impedir a reentrância, activando-a posteriormente ao concluir o processamento.
    - Tem a vantagem de desactivar apenas uma determinada interrupção, enquanto permite que outras interrupções possam ser processadas.



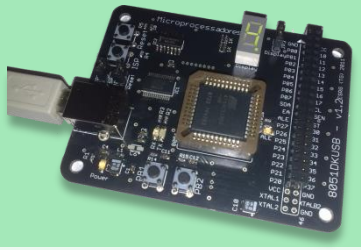
Interrupções: *O que são?*

- O que se entende por estado do CPU?
  - Na maioria dos processadores actuais, o estado do CPU é constituído pelos registos genéricos e pelas *flags* do processador
    - Normalmente, as *flags* do processador são salvaguardadas na pilha juntamente com o endereço de retorno ao código interrompido
    - A instrução retorno-da-interrupção executada no final da ISR restaura automaticamente as *flags* e transfere o controlo à aplicação interrompida
  - Ao seleccionar um processador deve-se compreender o que constituí o estado do CPU, bem como se as ISRs devem preservar algo para além dos registos genéricos



Interrupções: *O que são?*

- Porque é que uma ISR deve preservar o estado do CPU ?
  - Se não preservar o estado do CPU, quando a ISR termina a execução e devolve o controlo à aplicação interrompida, esta encontrará valores diferentes nos registos.
    - Repare que a aplicação não tem conhecimento de que foi interrompida.
    - Em tais casos, a depuração da aplicação é difícil, porque o sistema começará a "crashar" de forma estranha e de difícil reprodução – o cenário resultante do *crash* será único
      - Notar que cada vez que a ISR é executada e corrompe os registos, uma secção diferente do código da aplicação estará em execução



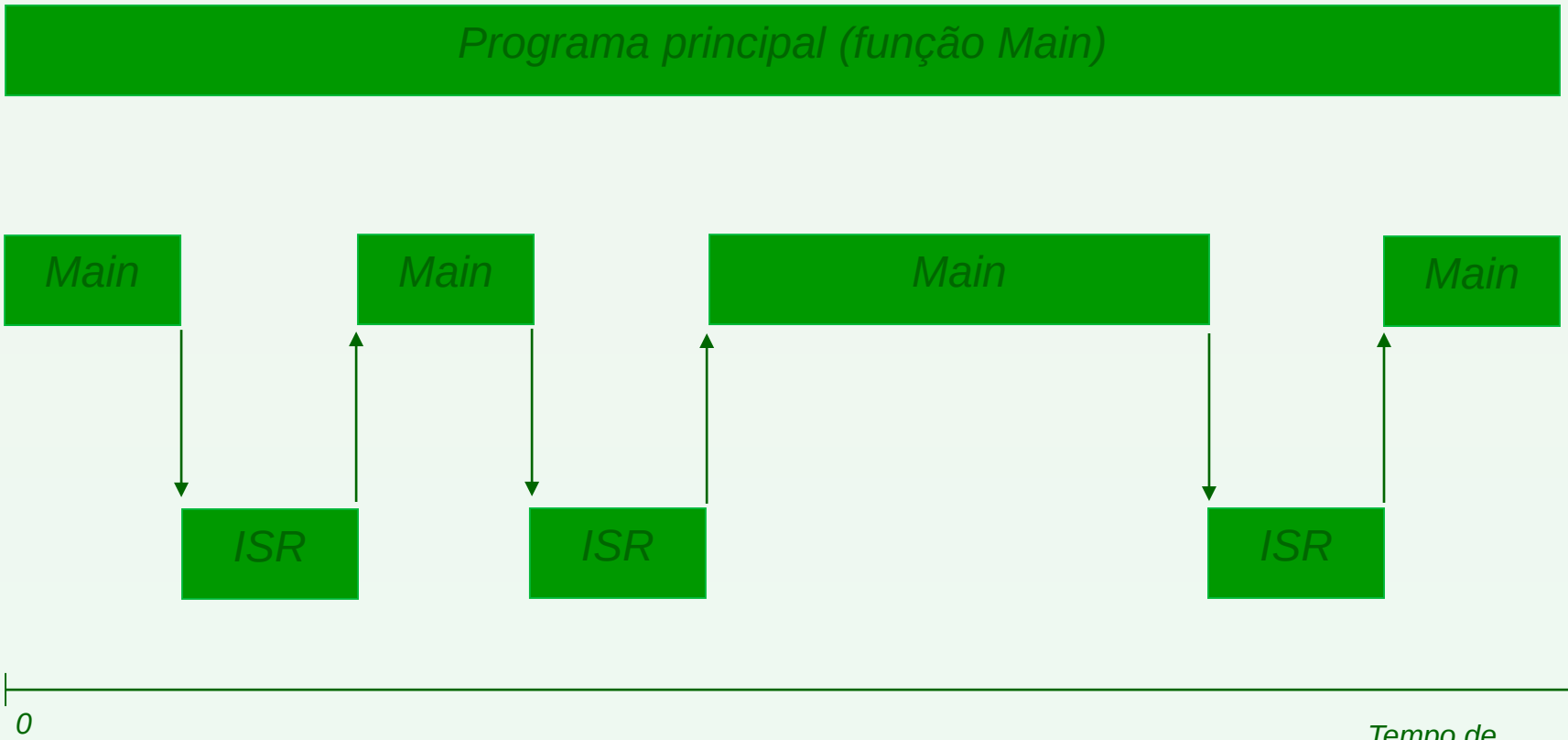
Interrupções no 8051

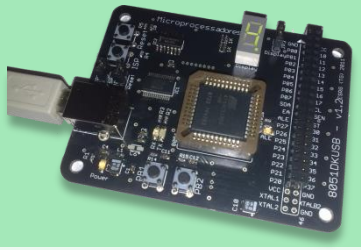
- **Ocorrência de uma interrupção aceite pelo CPU?**

ISR = Rotina de Serviço à Interrupção

Retorno da ISR após execução da instrução RETI

Ocorrência de um evento





Interrupções no 8051

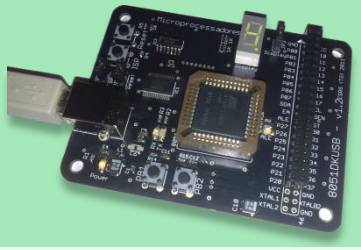
1. A instrução actualmente em execução será executada até ao fim
2. O PC é salvaguardado na pilha
3. O estado actual da interrupção é guardado internamente
4. O PC é carregado com o endereço do vector da ISR
5. ISR é executada

A ISR é concluída com a instrução RETI que:

i. Restaura o PC a partir da pilha
ii. Restaura o estado das interrupções
iii. Retoma a execução a partir do ponto onde foi interrompido

RETI

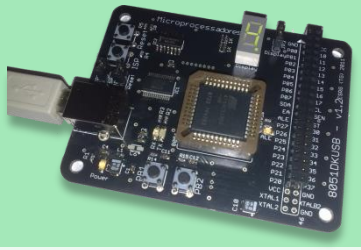
Bytes:	*1*
Cycles:	*2*
Encoding:	*00110010*
Operation:	*(PC15-8) ←((SP))*
	(SP) ← (SP) – 1
	(PC7-0) ←((SP))
	(SP) ← (SP) – 1



Interrupções no 8051

- **O 8051 possui 5 tipos de interrupções mais o RESET**
  - **Duas externas**
  - **Uma para a porta série**
  - **Duas para as unidades contadoras/temporizadoras 0 e 1 (*timers*)**
  - **Outros membros da família MCS-51 podem apresentar outras interrupções**

    Por exemplo, o 8052 apresenta uma interrupção extra dedicada ao *contador/temporizador 2*

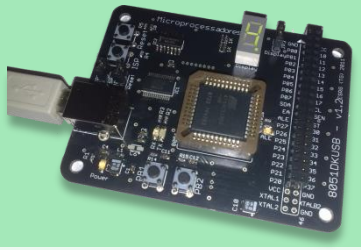


Interrupções no 8051

- **Tabela de Interrupções**
  - **Com vectores de interrupção que vão de 0 a 4BH (AT89C51RD2)**

    Para evitar a sobreposição de código sequencial sobre a área destinada aos vectores de interrupção, uma boa prática consiste em colocar um **JMP 050h** no endereço do RESET
  - **Existe um espaço de apenas 8 bytes entre os endereços de interrupção**
    i. 03h, 0Bh, 13h, 1Bh, 23h, 2Bh
    ii. O RESET começa no endereço 0000h
    iii. Caso o código de uma ISR for superior a 8 *bytes* deve-se colocar uma instrução de salto para uma subrotina no endereço do vector da interrupção
    iv. Interrupções: PCA (33H); Keyboard (3BH); SPI (4BH)
  - **O código da interrupção é executada a partir do endereço da interrupção**

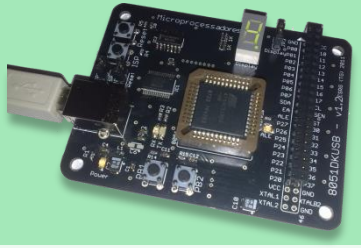
    i.e., o CPU espera encontrar código executável no endereço da interrupção



Interrupções no 8051

- **Tabela de Interrupções**

Interrupção	*Flag* que gera Interrupção	Bit no SFR	Endereço da ISR
RESET	RST		0000h
Externa 0	IE0	TCON.1	0003h
Timer 0	TF0	TCON.5	000Bh
Externa 1	IE1	TCON.3	0013h
Timer 1	TF1	TCON.7	001Bh
Porta Série	RI ou TI	SCON.0 ou SCON.1	0023h
Timer 2	TF2 ou EXF2	T2CON.7 ou T2CON.6	002Bh



Interrupções no 8051

- Ao entrar na ISR, a *flag* que causou a interrupção é automaticamente limpa pelo *hardware*

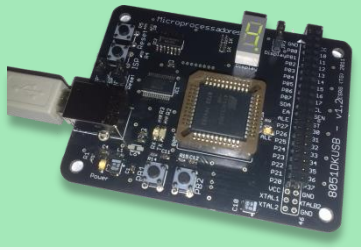
  Contudo, no caso das interrupções com origem em várias fontes, tais como a interrupção da porta série e interrupção do *Timer 2* a *flag* deve ser limpa pela ISR após determinar a causa da mesma

- Todas as interrupções são desactivadas após a inicialização (*Reset*) do sistema, pelo que devem ser activadas individualmente por *software*

- Ao codificar uma ISR deve-se garantir a inexistência de interacções indesejáveis entre o programa interrompido e a ISR

  1. A garantia é dada pela preservação dos registos no instante da comutação de contexto

  2. No caso de se comutar de banco de registos, deve-se também alterar o apontador da pilha (SP)

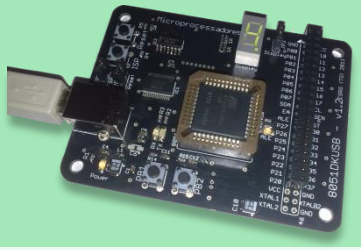


Interrupções no 8051

- **Regra geral, a ISR deve proteger os seguintes registos:**
  - **PSW**
  - **DPTR (DPH/DPL)**
  - **ACC**
  - **B**
  - **R0-R7**

```
            CSEG    AT 3h       ; endereço do vector interrupção externa 0
            LJMP    ISR_EXT0

            CSEG    AT 1000h
ISR_EXT0:   PUSH    PSW         ; salvaguarda as flags
            MOV     PSW, #00001000b ; selecciona o banco de registo 1
            MOV     SP, #??     ; alterar aqui o apontador da pilha (SP)
            ...
            POP     PSW         ; restaura as flags
            RETI
```

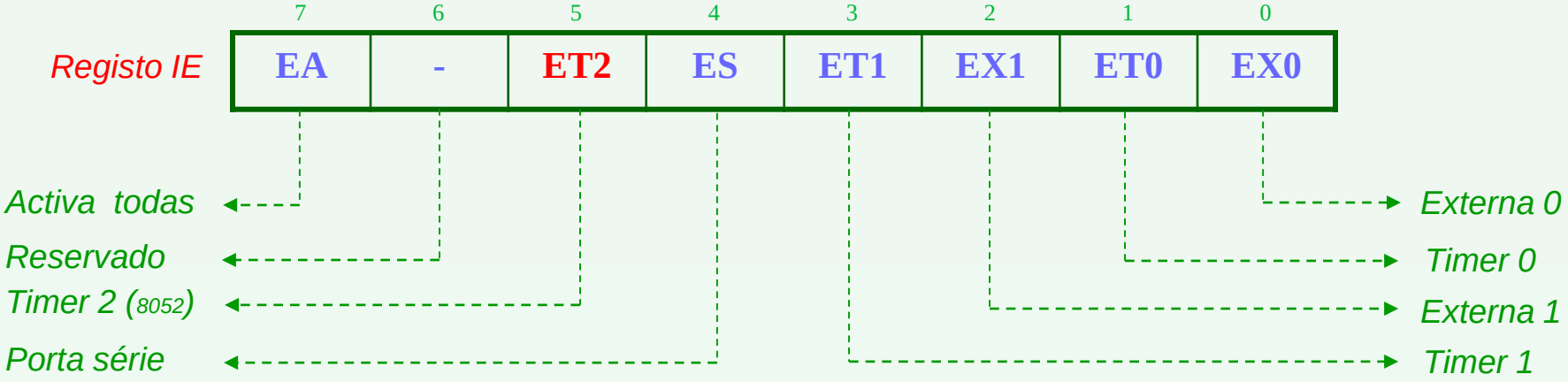


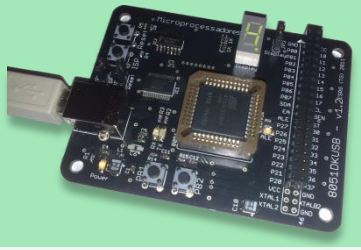
Interrupções no 8051

- **Activação e desactivação das interrupções**
  - O programa controla o processamento de uma interrupção a dois níveis, através da manipulação dos bits do registo IE:

    i. **O primeiro nível é controlado pelo bit EA**
    - EA = 0 (CLR EA) todas as interrupções são ignoradas
    - EA = 1 (SETB EA) o segundo nível será considerado

    ii. **O segundo nível é controlado individualmente para cada interrupção**
    - As interrupções são activadas colocando o respectivo bit a 1



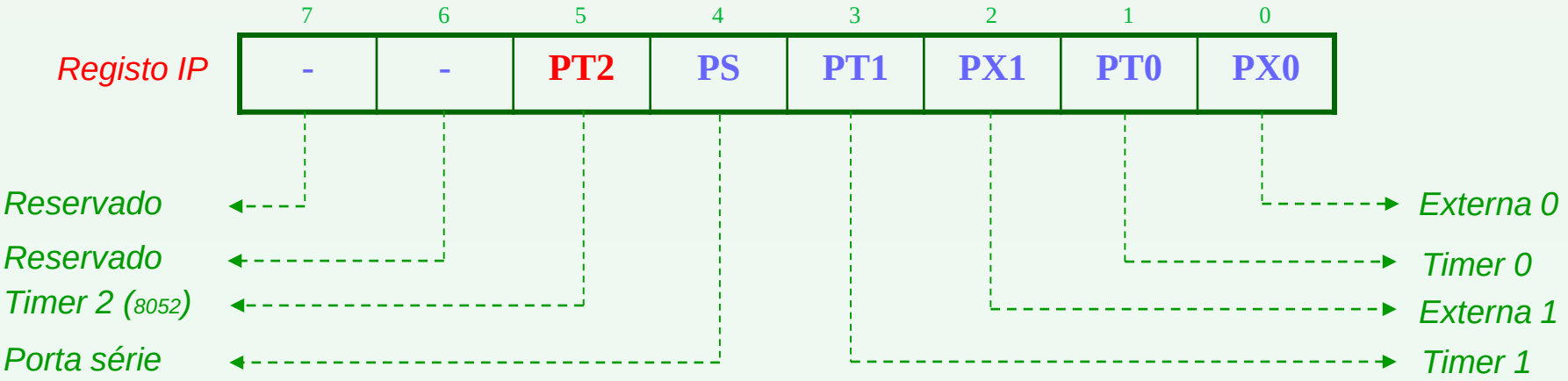


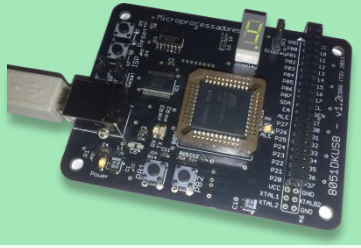
Interrupções no 8051

- **Prioridade das Interrupções**
  - **Cada interrupção pode ter dois níveis de prioridade**

    Os níveis de prioridade são programados através do registo IP
  - **Uma interrupção apenas pode interromper outra de prioridade mais baixa**
  - **Na ocorrência simultânea de duas interrupções com mesmo nível de prioridade, o CPU será cedido à ISR seleccionada pela seguinte sequência interna de *polling*:**

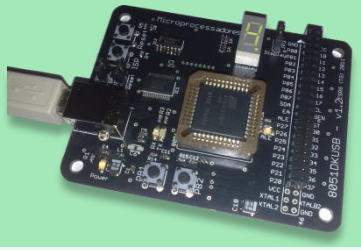
    1º externa 0, 2º *timer* 0, 3º externa 1, 4º *timer* 1, 5º porta série e 6º *timer* 2





Interrupções AT89C51RD2

Number	Polling Priority	Interrupt Source	Interrupt Request	Vector Address
0	0	Reset		0000h
1	1	INT0	IE0	0003h
2	2	Timer 0	TF0	000Bh
3	3	INT1	IE1	0013h
4	4	Timer 1	IF1	001Bh
5	6	UART	RI+TI	0023h
6	7	Timer 2	TF2+EXF2	002Bh
7	5	PCA	CF + CCFn (n = 0 - 4)	0033h
8	8	Keyboard	KBDIT	003Bh
9	9	-	-	0043h
10	10	SPI	SPIIT	004Bh



Interrupções AT89C51RD2

Table 17-3. IEN0 Register

IEN0 - Interrupt Enable Register (A8h)

7	6	5	4	3	2	1	0
EA	EC	ET2	ES	ET1	EX1	ET0	EX0

Bit Number	Bit Mnemonic	Description
7	EA	**Enable All interrupt bit** Cleared to disable all interrupts. Set to enable all interrupts.
6	EC	**PCA interrupt enable bit** Cleared to disable. Set to enable.
5	ET2	**Timer 2 overflow interrupt Enable bit** Cleared to disable timer 2 overflow interrupt. Set to enable timer 2 overflow interrupt.
4	ES	**Serial port Enable bit** Cleared to disable serial port interrupt. Set to enable serial port interrupt.
3	ET1	**Timer 1 overflow interrupt Enable bit** Cleared to disable timer 1 overflow interrupt. Set to enable timer 1 overflow interrupt.
2	EX1	**External interrupt 1 Enable bit** Cleared to disable external interrupt 1. Set to enable external interrupt 1.
1	ET0	**Timer 0 overflow interrupt Enable bit** Cleared to disable timer 0 overflow interrupt. Set to enable timer 0 overflow interrupt.
0	EX0	**External interrupt 0 Enable bit** Cleared to disable external interrupt 0. Set to enable external interrupt 0.

Reset Value = 0000 0000b
Bit addressable

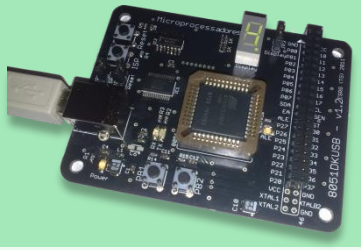
Table 17-6. IEN1 Register

IEN1 - Interrupt Enable Register (B1h)

7	6	5	4	3	2	1	0
-	-	-	-	-	ESPI	-	KBD

Bit Number	Bit Mnemonic	Description
7	-	**Reserved**
6	-	**Reserved**
5	-	**Reserved**
4	-	**Reserved**
3	-	**Reserved**
2	ESPI	**SPI interrupt Enable bit** Cleared to disable SPI interrupt. Set to enable SPI interrupt.
1		**Reserved**
0	KBD	**Keyboard interrupt Enable bit** Cleared to disable keyboard interrupt. Set to enable keyboard interrupt.

Reset Value = XXXX X000b
Bit addressable



Interrupções AT89C51RD2

Table 17-4. IPL0 Register

IPL0 - Interrupt Priority Register (B8h)

7	6	5	4	3	2	1	0
-	PPCL	PT2L	PSL	PT1L	PX1L	PT0L	PX0L

Bit Number	Bit Mnemonic	Description
7	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit.
6	PPCL	**PCA interrupt Priority bit** Refer to PPCH for priority level.
5	PT2L	**Timer 2 overflow interrupt Priority bit** Refer to PT2H for priority level.
4	PSL	**Serial port Priority bit** Refer to PSH for priority level.
3	PT1L	**Timer 1 overflow interrupt Priority bit** Refer to PT1H for priority level.
2	PX1L	**External interrupt 1 Priority bit** Refer to PX1H for priority level.
1	PT0L	**Timer 0 overflow interrupt Priority bit** Refer to PT0H for priority level.
0	PX0L	**External interrupt 0 Priority bit** Refer to PX0H for priority level.

Reset Value = X000 0000b
Bit addressable

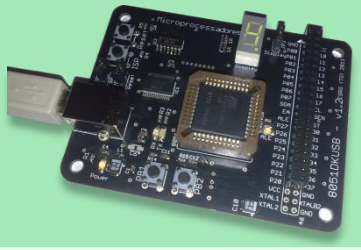
Table 17-5. IPH0 Register

IPH0 - Interrupt Priority High Register (B7h)

7	6	5	4	3	2	1	0
-	PPCH	PT2H	PSH	PT1H	PX1H	PT0H	PX0H

Bit Number	Bit Mnemonic	Description
7	-	**Reserved** The value read from this bit is indeterminate. Do not set this bit.
6	PPCH	**PCA interrupt Priority high bit.** PPCH PPCL Priority Level 0 0 Lowest 0 1 1 0 1 1 Highest
5	PT2H	**Timer 2 overflow interrupt Priority High bit** PT2H PT2L Priority Level 0 0 Lowest 0 1 1 0 1 1 Highest
4	PSH	**Serial port Priority High bit** PSH PSL Priority Level 0 0 Lowest 0 1 1 0 1 1 Highest
3	PT1H	**Timer 1 overflow interrupt Priority High bit** PT1H PT1L Priority Level 0 0 Lowest 0 1 1 0 1 1 Highest
2	PX1H	**External interrupt 1 Priority High bit** PX1H PX1L Priority Level 0 0 Lowest 0 1 1 0 1 1 Highest
1	PT0H	**Timer 0 overflow interrupt Priority High bit** PT0H PT0L Priority Level 0 0 Lowest 0 1 1 0 1 1 Highest
0	PX0H	**External interrupt 0 Priority High bit** PX0H PX0L Priority Level 0 0 Lowest 0 1 1 0 1 1 Highest

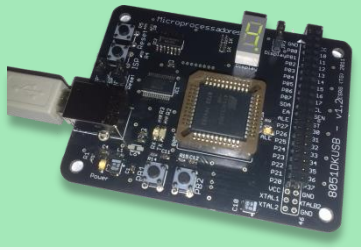
Reset Value = X000 0000b
Not bit addressable



Interrupções no 8051

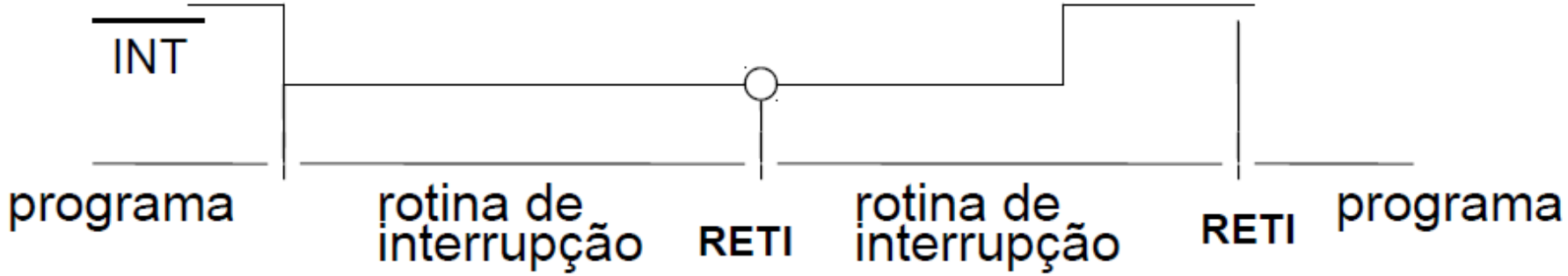
- **Interrupções Externas: INT0 e INT1**
  - **Pinos de entrada: /INT0 (P3.2) e /INT1(P3.3) devem ser inicializadas a 1; *(Porquê ?)***
  - **As interrupções externas podem ser programadas através dos bits IT0 (TCON.0) e IT1 (TCON.2) para funcionar por:**
    i. **Nível (ITX=0)**
    ii. **Transição descendente (ITX = 1)**
  - **Os pinos das interrupções externas P3.2 e P3.3 são amostrados uma única vez por cada ciclo de execução (*fetch-decode-execute*)**

A entrada deve manter-se inalterada pelo menos 12 períodos de relógio para garantir a activação da *flag* de interrupção (EX0 ou EX1)

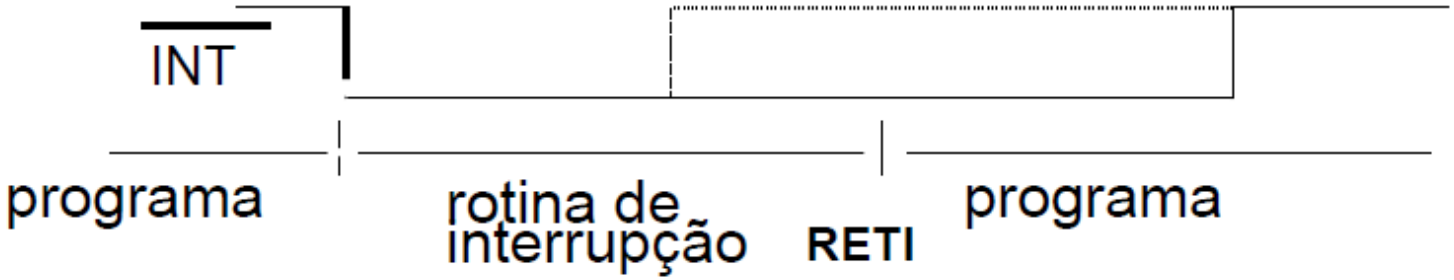


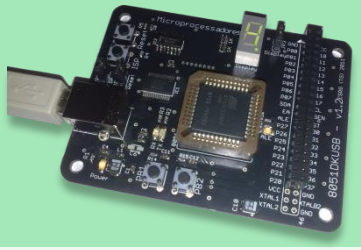
Interrupções externas

- Sensível ao nível



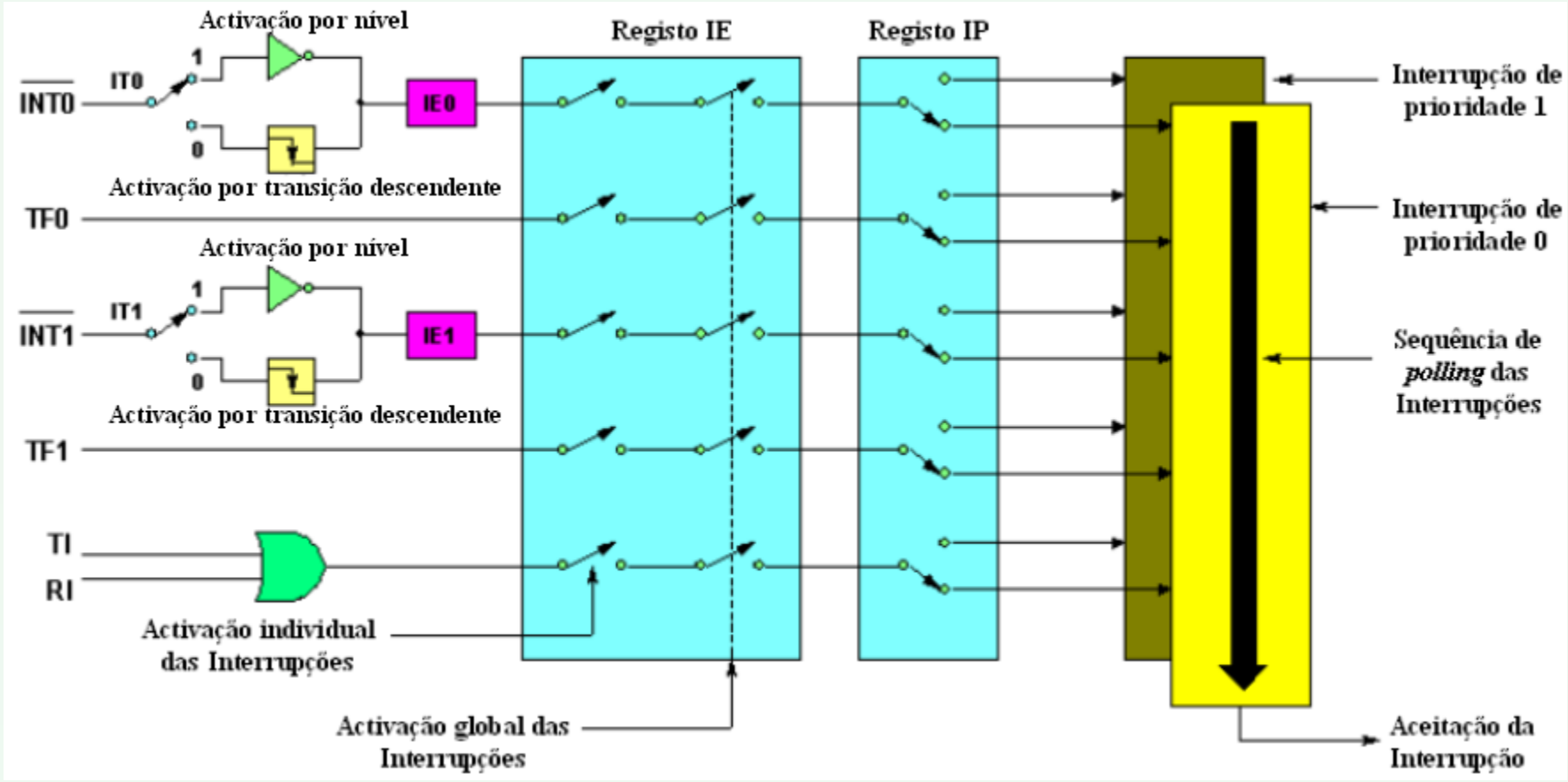
- Sensível à transição





Interrupções no 8051

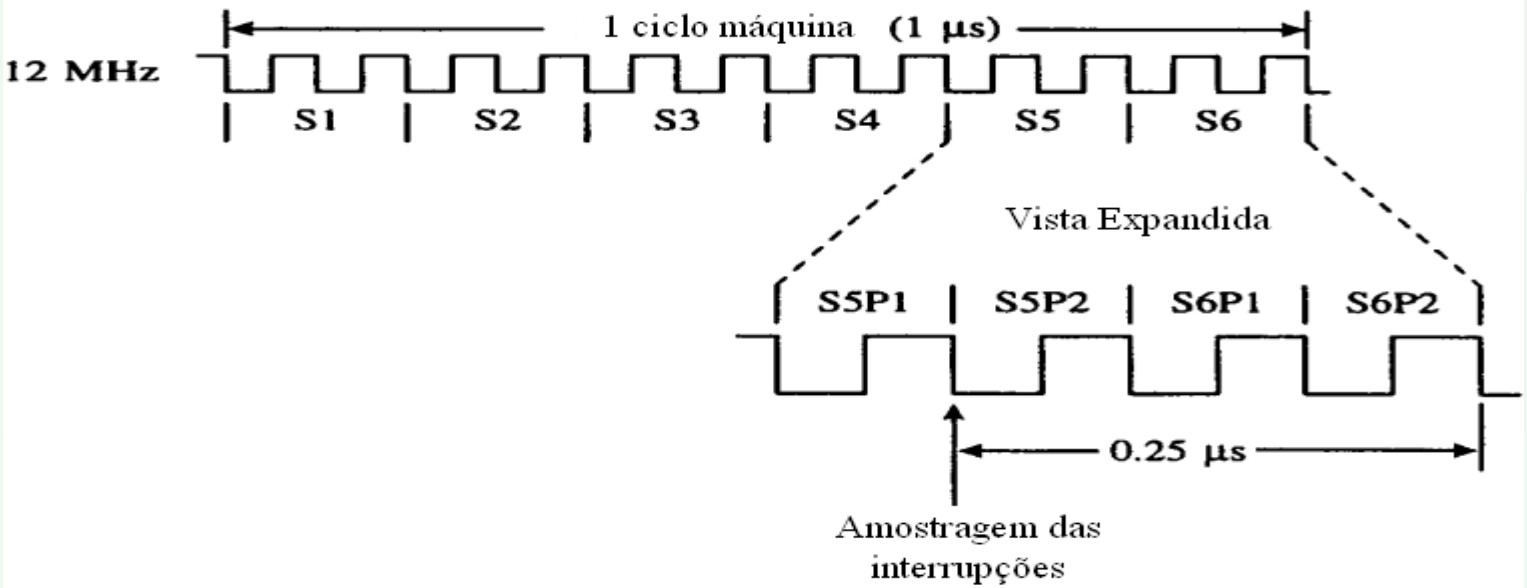
- **Estrutura genérica da interrupção no 8051**



Interrupções no 8051

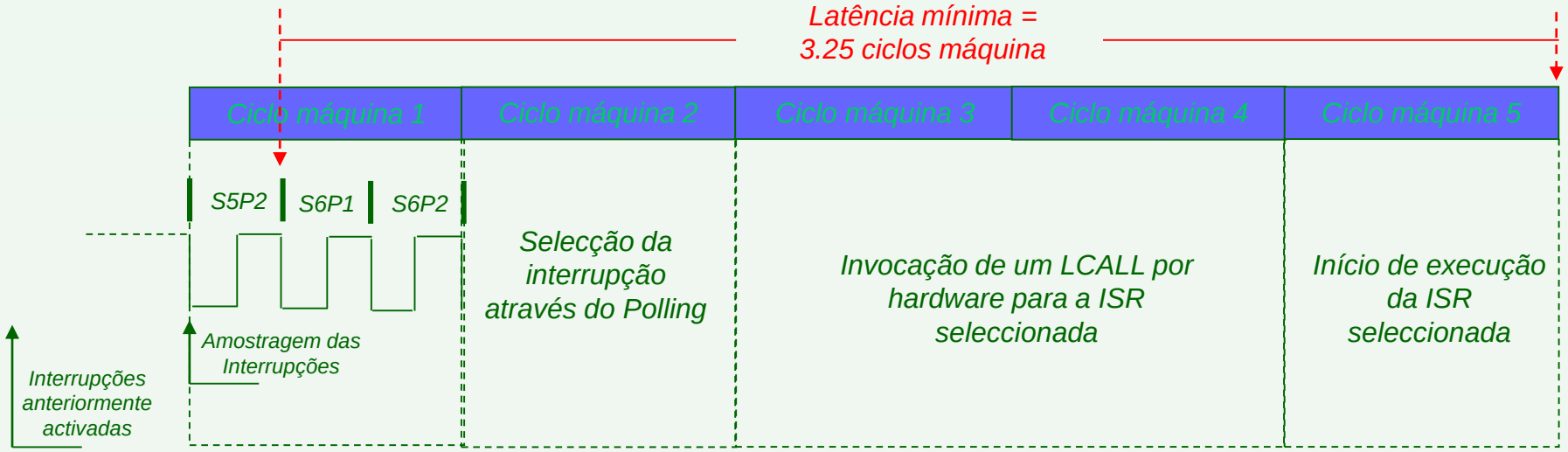
- **Temporizações da Interrupção**
  - **Cada ciclo máquina consiste de 12 ciclos de relógio**
  - **Cada ciclo máquina está dividida em 6 estados (S1 – S6) e cada estado pode ser decomposto em duas fases (SxP1 e SxP2)**

    Cada estado tem a duração de dois ciclos de relógio
  - **As flags das interrupções são amostradas e guardadas na segunda fase do quinto estado (S5P2)**



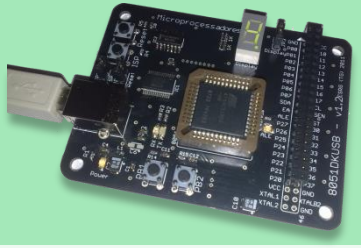
Interrupções no 8051

- **Sequência de atendimento de uma interrupção**



- **O que entende por Latência da Interrupção?**

  A latência de uma interrupção é o tempo decorrido entre o instante de emissão de uma interrupção e o instante da execução da primeira instrução da ISR associada à interrupção



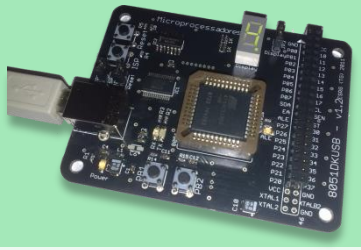
Interrupções no 8051

- **Após a amostragem, no próximo ciclo máquina as amostras são submetidas ao *polling* e estando uma *flag* activa, a interrupção será aceite caso forem satisfeitas todas as seguintes condições:**

  1. **Não está em execução nenhuma ISR de prioridade superior ou igual**
  2. **Foi concluída a execução da instrução processada no ciclo de *polling* actual**
  3. **A instrução em execução não é um RETI ou um acesso aos registos IE e IP**

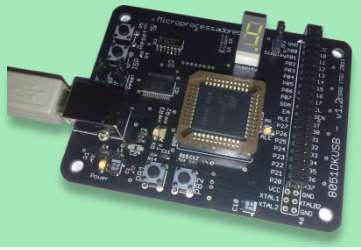
- **Nos dois ciclos máquina seguintes, a instrução de retorno será colocado na pilha e PC carregado com o endereço do vector da interrupção seleccionada**

A partir do próximo ciclo a ISR entra em execução



Interrupções no 8051

- **Resumindo:**
  - **Para usar qualquer interrupção do 8051, devem ser efectuados os seguintes passos:**
    1. **Colocar a 1 o bit EA do registo IE**
    2. **Colocar a 1 o bit de activação individual da interrupção em questão (outro bit do registo IE)**
    3. **Iniciar a ISR no endereço do vector associado à interrupção em questão**
    4. **No caso das interrupções externas deve-se ainda**
       i. **Inicializar adequadamente os bits IT0 e/ou IT1**
       ii. **Colocar a 1 os pinos INT0 (P3.2) e/ou INT1 (P3.3)**

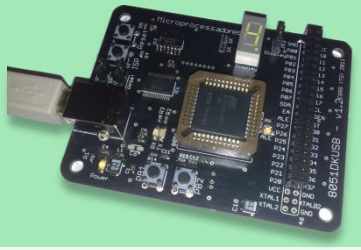


Interrupções no 8051

- **Exemplo de um programa com interrupção**

```
CSEG    AT 0
JMP     MAIN
;vector de interrupção do timer 0
CSEG    AT 0Bh      ; endereço do vector timer0
;Por Fazer: Inserir aqui o código da ISR caso < 8 bytes
...
RETI                ; retorno da interrupção do timer 0
;vector de interrupção da porta série
CSEG    AT 23h      ; endereço do vector da porta série
LJMP    SerISR;     ; caso o código da ISR > 8bytes
...
CSEG    AT 33h      ; fim da área das interrupções
MAIN:
```

```
; Configuração global e individual das interrupções a
; usar. Neste caso a do timer 0  e da porta série
;assim como outras configurações, como por
;exemplo  a do timer
SETB EA
...
; Inserir aqui código para processar algo
...
JMP     $       ; aguarde pelas interrupções
SerISR:         ; ISR da porta série
        ...
        RETI
END
```



Interrupções no 8051

```
JNB  TF0, SKIP
CPL  P3.0
CLR TF0
DO_IT:  ADD A, #3
SKIP:
        ...
```

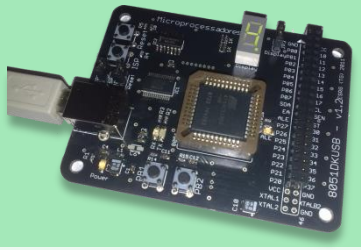
100 Ciclos a cada iteração

1. Estando o Timer 0 configurado no modo 16-bits, a cada 65.536 ciclos máquinas acontecerá um overflow
   i. Durante esse tempo, o JNB foi executado 655 vezes: equivale a 1310 ciclos máquina
2. Além do código ser visualmente "feio" por ser necessário efectuar o teste a cada iteração, gasta-se aproximadamente 2% do tempo a verificar quando comutar o estado do pino P3.0

Usando a interrupção deixa de ser necessário efectuar o teste da condição

1. O μP realiza de forma automática o teste e caso se verifique a condição, invoca uma ISR
2. Deixa de ser necessário a execução da instrução CLR TF0 porque o 8051 ao entrar na ISR "limpa" automaticamente esta flag

```
ISR:  CPL     P3.0
      RETI
```



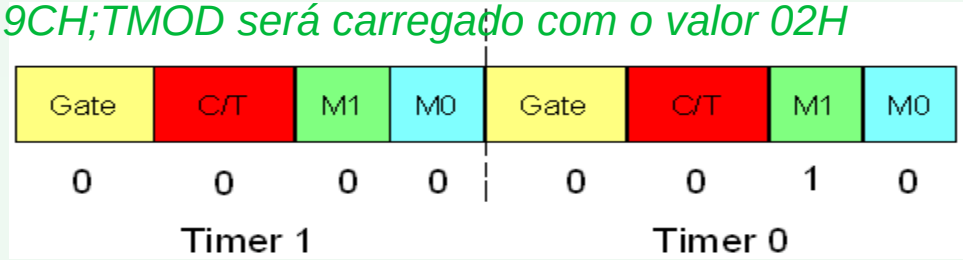
Interrupções no 8051

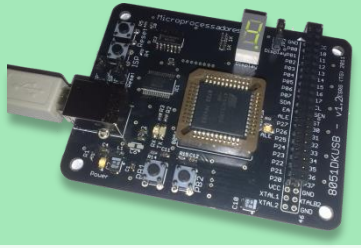
Exercício:

Escreva um programa que lê continuamente 8-bits de dados de P0 e envia-os para P1, e ao mesmo tempo gera uma onda quadrada com 5kHz de frequência no pino P2.1.

Análise do Problema:

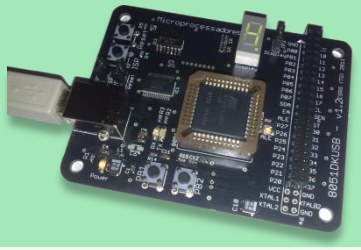
1. *O período da onda é 200μs = 1/5KHz*
   - *O tempo no nível 1 = tempo nível 0 =100 μs*
2. *Assumindo um cristal de 12MHz*
   - *Tempo de contagem = 100*
3. *Sendo este intervalo menor do que 256 pode-se usar o timer 0 no modo 2 (auto-reload)*
   - *TH0 = 256 – 100 = 156 = 9CH;TMOD será carregado com o valor 02H*





Interrupções no 8051

```
        CSEG    AT 0000H
        LJMP    MAIN            ; não sobrepor a tabela de vectores de interrupções
;ISR do Timer 0 usado para gerar a onda quadrada
        CSEG    AT 000BH        ; endereço do vector de interrupção do timer 0
        CPL     P2.1            ; comuta o estado do pino P2.1
        RETI                    ; Regressa da ISR (retorna ao programa principal)
; Código do programa principal a partir do endereço 0033h
        CSEG    AT 0033H        ; após o espaço reservado à tabela de vectores de interrupções
MAIN:   MOV     P0,#0FFH        ; Configure o P0 como porto de entrada
        MOV     TMOD,#02H       ; configurar Timer 0 no modo 2 (auto reload)
        MOV     TH0,#09CH       ; TH0 = 9CH = 100µ para uma frequência de 5KHz
        MOV     IE,#82H         ; IE = 10000010B activação global e a da interrupção do Timer 0
        SETB    TR0             ; arranca o Timer 0
LOOP:   MOV     A,P0            ; lê os 8-bits de dados a partir de P0
        MOV     P1,A            ; envia os 8-bits de dados lidos para P1
        SJMP    LOOP            ; continua a leitura a partir de P0 e envia para P1 a menos que seja
                                ; interrompido por TF0
        END
```



Prioridade das Interrupções

- Estrutura de prioridades das interrupções:
  - Mais um registo de prioridade IPH (B7H);
  - O que significa que há quatro níveis de prioridade;
  - O esquema de prioridades é idêntico ao do 8051:
    - Uma interrupção será atendida desde que uma outra interrupção do mesmo nível de prioridade ou superior não esteja a ser atendida. Caso contrário, espera pelo final da interrupção actual.

	7	6	5	4	3	2	1	0
IP (B8H)	-	PPC	PT2	PS	PT1	PX1	PT0	PX0

	7	6	5	4	3	2	1	0
IPH (B7H)	-	PPCH	PT2H	PSH	PT1H	PX1H	PT0H	PX0H

Bits de Prioridade		**Interrupção**
IPH.x	**IP.x**	**Nível de prioridade**
0	0	0 - Mais Baixa
0	1	1
1	0	2
1	1	3 - Mais Alta

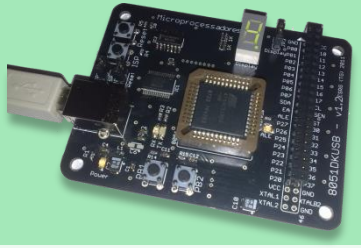


Tabela de Interrupções

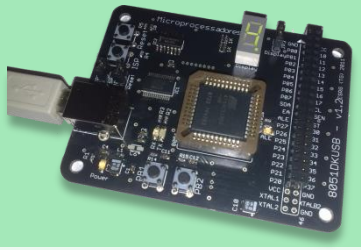
Fonte	Prioridade	Flags	Flag Limpa	Endereço do Vector
X0	1	IE0	Sim - Edge	03H
T0	2	TF0	Sim	0BH
X1	3	IE1	Sim - Edge	13H
T1	4	TF1	Sim	1BH
PCA	5	CF,CCFn	Não	33H
SP	6	RI,TI	Não	23H
T2	7	TF2,EXF2	Não	2BH

IE (A8H)

7	6	5	4	3	2	1	0
EA	EC	ET2	ES	ET1	EX1	ET0	EX0

Símbolo

EA	Permite(1)/Impede(0) todas as interrupções
EC	Bit de permissão do módulo PCA
ET2	Bit de permissão do Timer 2
ES	Bit de permissão do porto Série
ET1	Bit de permissão do Timer 1
EX1	Bit de permissão da interrupção Externa 1
ET0	Bit de permissão do Timer 0
EX0	Bit de permissão da interrupção Externa 0



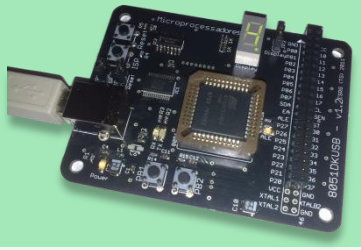
Registos extra

- Dois registos auxiliares extra:
  - AUXR e AUXR1;
- Permitem controlar algumas funções extra do AT89C51RD2.

	7	6	5	4	3	2	1	0
AUXR (8EH)	DPU	-	MO	XRS2	XRS1	XRS0	EXTR	AO

Símbolo

DPU	Desabilitar *weak pull-up* (poupar energia)
-	Reservado
MO	Duração do pulso de RD e WR (6 ou 12 clocks)
XRS2	Tamanho da XRAM: 000 - 256 bytes
XSR1	001 - 512 bytes ; 010 - 768 bytes (default)
XRS0	011 - 1024 bytes; 100 - 1792 bytes
EXTRAM	Quando a 0 o acesso é à XRAM *on-chip*. A 1 o acesso é normal
AO	Bit de permissão da interrupção Externa 0



Duplo DPTR

	7	6	5	4	3	2	1	0
AUXR1	-	-	ENB	-	GF2	0	-	DPS

A2H

ENB Enable Boot Loader

Automaticamente a 1 se PSEN=0

GF2 Flag de propósito geral

DPS Selecção de DPTR

- As instruções sobre o DPTR referem-se ao DPTR que está actualmente seleccionado usando o registo AUXR1 (bit 0). Há seis instruções que usam o DPTR:

  - INC DPTR
  - MOVC A,@A+DPTR
  - MOVX @DPTR,A
  - MOV DPTR,#data16
  - MOVX A,@DPTR
  - JMP @A+DPTR

Ver Application Note AN458 para detalhes sobre o acesso a DPH e DPL de cada DPTR

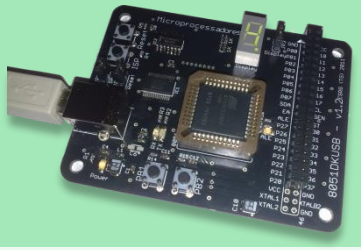


Tabela de saltos

- JMP @A+DPTR
  - O salto é feito para o endereço dado por A+DPTR;
  - Como as instruções de salto são AJMP (2 bytes) ou LJMP (3 bytes), o acumulador tem de ser multiplicado por dois ou por três:

```
SALTOS:     MOV     DPTR,#TABSALTOS
            RL      A                   ;multiplica índice por 2
            JMP     @A+DPTR
….
TABSALTOS:
            AJMP    ROT0
            AJMP    ROT1
            AJMP    ROT2
            ….
            AJMP    ROTN
```