Прекъсвания и директен достъп до паметта

Автор: гл. ас. д-р инж. Любомир Богданов

Европейски съюз

ПРОЕКТ BG051PO001--4.3.04-0042
„Организационна и технологична инфраструктура за учене през целия живот и развитие на компетенции”
Проектът се осъществява с финансовата подкрепа на
Оперативна програма „Развитие на човешките ресурси”,
съфинансирана от Европейския социален фонд на Европейския съюз
Инвестира във вашето бъдеще!

ЕСФ България
Европейски социален фонд

Съдържание

1. Обслужване на прекъсвания
2. Контролер за директен достъп
3. Методи за понижаване на E/P
4. Схеми за генериране на тактов сигнал

Обслужване на прекъсвания

Прекъсване (interrupt) е процес, при който микропроцесорът спира изпълнението на главната програма и започва да изпълнява кода на друга програма, в следствие на хардуерно събитие.

Това събитие може да е **синхронно** или **асинхронно** на изпълнението на главната програма.

С помощта на прекъсванията се премахва нуждата от **постоянна проверка** дали дадено събитие е настъпило (метод "**polling**"), което е **излишно изразходване на изчислителен ресурс**.

Обслужване на прекъсвания

Хендлер на прекъсване (interrupt handler) - допълнителна програма, различна от основната main, която се изпълнява вследствие на прекъсване. Нейната цел е да обслужи прекъсването, т.е. да се извършат дейности в отговор на постъпилото прекъсване.

Вектор на прекъсване (interrupt vector) - адресът от паметта, на който се намира кодът, изпълняван при настъпило прекъсване. От софтуерна гледна точка това е указател към функцията (хендлерът), която се извиква при настъпило събитие.

Обслужване на прекъсвания

Векторна таблица (interrupt vector table) - масив от указатели към функции, който се използва за обслужване прекъсванията. Всеки елемент от този масив е вектор на прекъсване и при настъпване на събитие се извлича един от много хендлери на прекъсване по хардуерен път.

Приоритет на прекъсване (interrupt priority) – при настъпването на две или повече събития се налага приоритизиране на извикването на хендлерите, защото микропроцесорът може да изпълнява само една програма в даден момент. Прекъсването с по-висок приоритет ще се обслужи преди прекъсването с по-нисък приоритет.

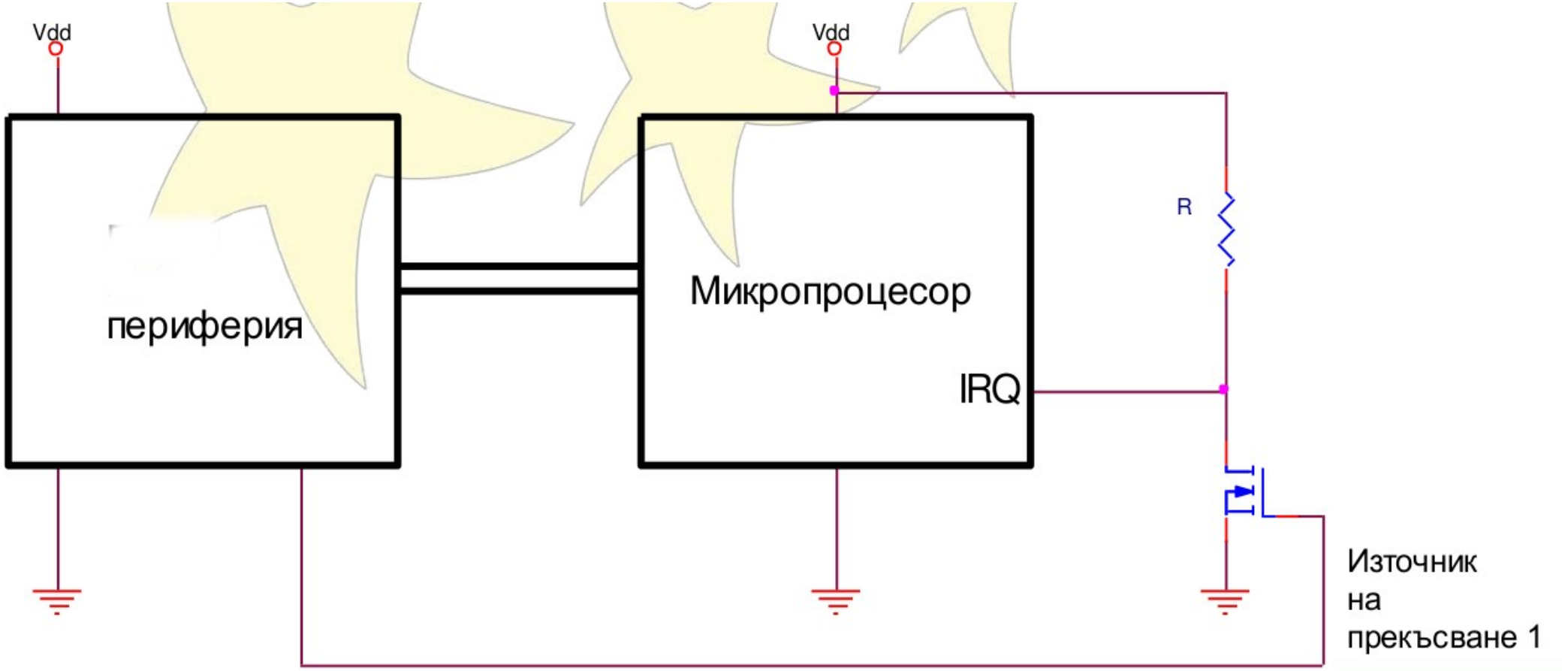
Обслужване на прекъсвания

На фигурата на следващия слайд е демонстриран пример за реализация на прекъсване от един източник. Както се вижда микропроцесорът превключва от изпълнението на главната програма в изпълнение на хендлера на прекъсване. Източник на сигнал IRQ може да е периферно устройство, което след извършване на дадена функция да сигнализира на микропроцесора за събитието чрез прекъсване.

Броят на IRQ входовете се определя от броя на микропроцесорните ядра. В многоядрените системи софтуерът трябва да определи кое прекъсване от кое ядро да се обслужи (в частност – това се прави от операционната система).

Забележка: суперскаларен не означава многоядрен!

Обслужване на прекъсвания

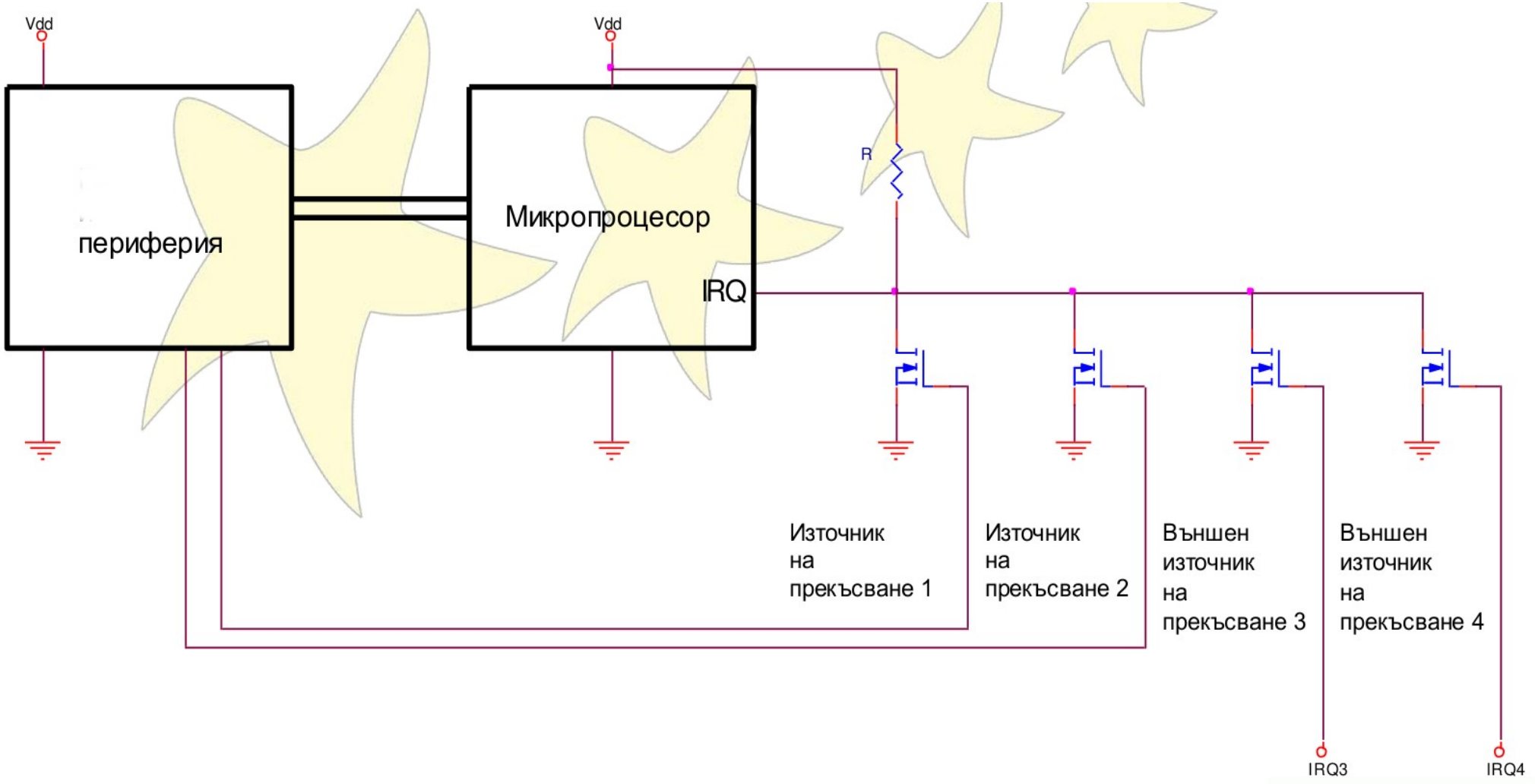


Обслужване на прекъсвания

В практиката по-често срещания вариант е получаване на прекъсване от два или повече източника, както е показано на фигурата. Транзисторите образуват логическата функция **жично ИЛИ**.

В този случай микропроцесорът се нуждае от допълнителен механизъм за **разпознаване на източника**.

Обслужване на прекъсвания



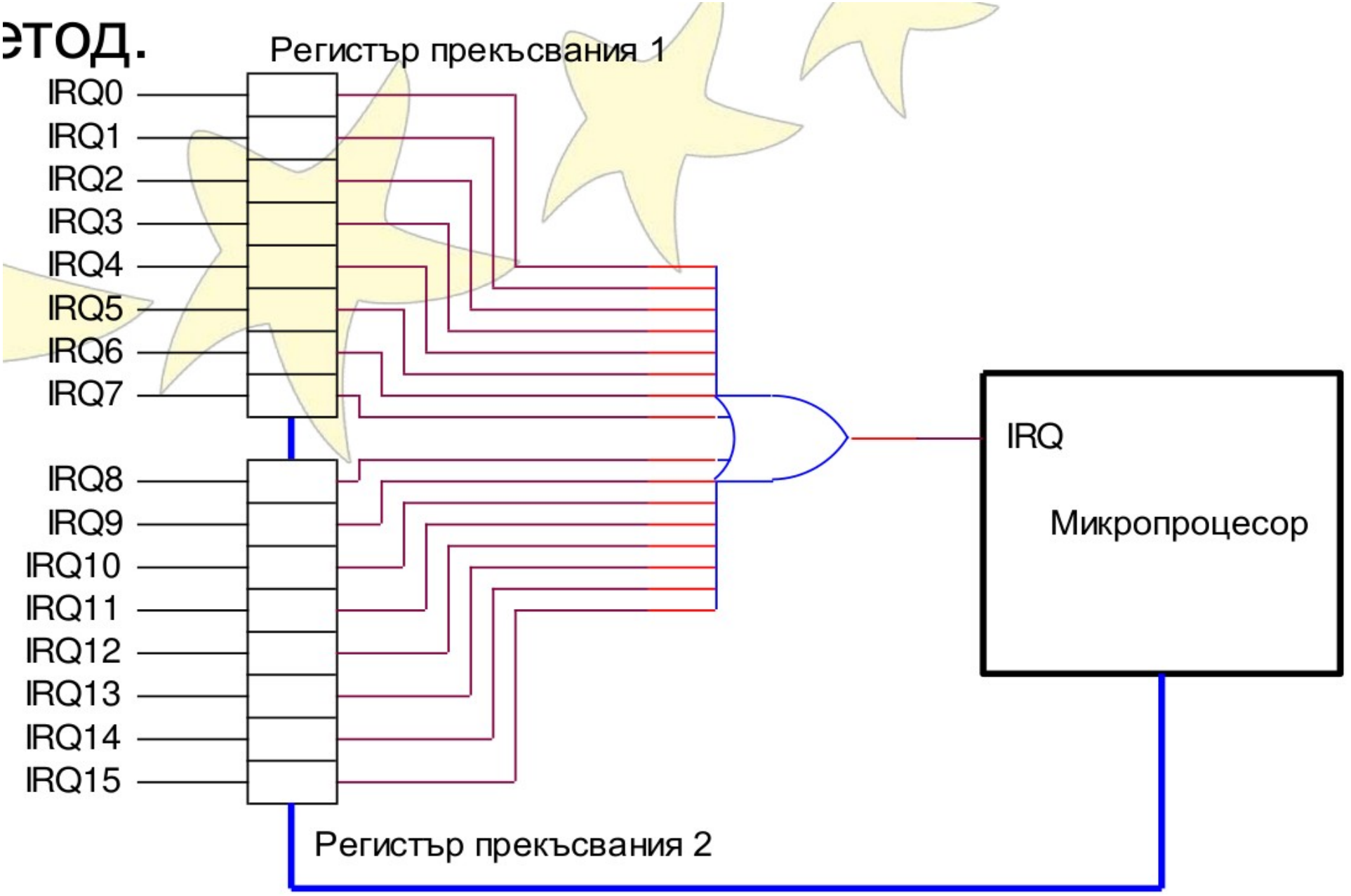
Обслужване на прекъсвания

За да може микропроцесорът да разбере от кой точно източник е получена заявката за прекъсване, **сигналите се буферират в един или няколко регистъра**, които са **част от адресното поле**.

След получаване на сигнал за прекъсване микропроцесорът прочита тези регистри и, знаейки тяхната предишна стойност, определя източника.

На фигурата на следващия слайд е демонстриран този метод.

Обслужване на прекъсвания



Обслужване на прекъсвания

Прекъсванията могат да се разделят на два вида – маскируеми и немаскируеми.

Маскируеми прекъсвания – прекъсвания, които могат да бъдат забранявани. Ако дадено прекъсване е забранено, при появата на сигнал от източника няма да се подаде сигнал към микропроцесора.

Пример – приети данни в преместващия регистър на UART модула.

Пример – преобразуването с АЦП е завършило.

Обслужване на прекъсвания

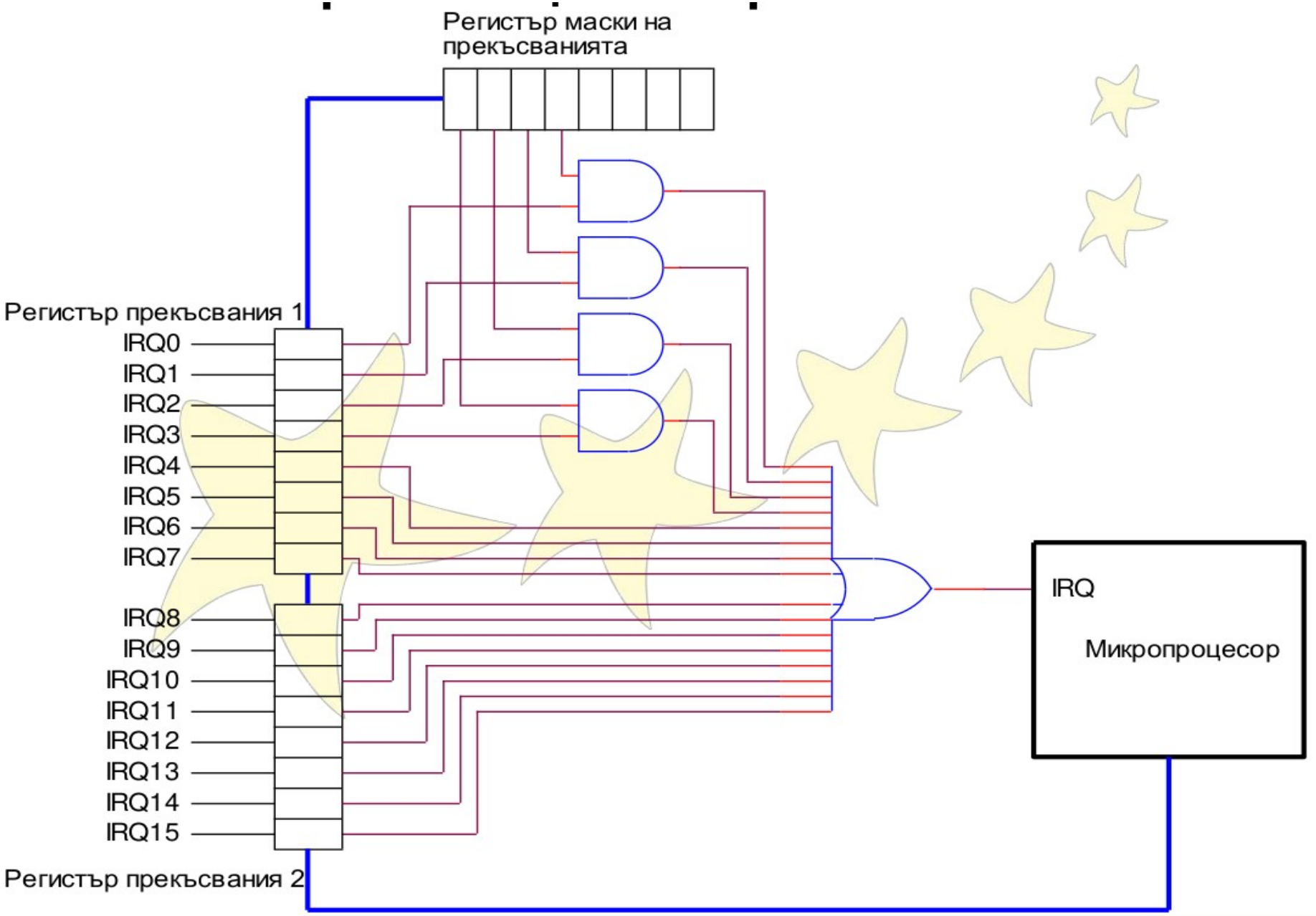
Немаскируеми прекъсвания – прекъсвания, които винаги трябва да се обслужват.

Пример - RESET сигнала – микропроцесорът ще бъде рестартиран винаги при наличието на активно ниво на този сигнал.

Пример - прекъсване при прегряване на чипа.

На следващия слайд е показан пример с 4 маскируеми прекъсвания.

Обслужване на прекъсвания



Обслужване на прекъсвания

Етапите, през които преминава микропроцесорът при поява на заявка за прекъсване са [1] [2]:
*μPU копира съдържанието на **стековата група** (stack frame) в стека, т.е. някъде около края на SRAM при намаляващ стек, или някъде около началото на SRAM при растящ стек. **Това става хардуерно**, т.е. никъде в кода няма да се видят асемблерни инструкции, които да копират регистри.

Пример – на MSP430 стековата група е съставена от PC, SR.
Пример - ARM Cortex-M стековата група е съставена от PSR, PC, R14, R12, R3, R2, R1, R0.

Обслужване на прекъсвания

За инструкции с числа с плаваща запетая, виж лекцията за FPU.

*паралелно с PUSH-ването на стековата група се **прочита адреса на хендлера** от векторната таблица и се записва в програмния брояч (PC). Това е еквивалентно на влизане в хендлера.
***изпълнява се хендлера** на прекъсването. Добре-написаният фърмуер има много малко код в хендлерите. В хендлерите трябва да се записват флагове, които да сигнализират на main( ) функцията, че събитието се е случило. Обработката на събитието е добре да се случи в main( ).

Времезакъснения (delay) и извеждането на дебъг съобщения (printf) трябва да се избягват в хендлерите.

Обслужване на прекъсвания

Ако в хендлера се използват регистри от ядрото, **които не са част от стековата група**, те трябва да бъдат копирани чрез асемблерни инструкции (PUSH) на стека (SRAM) в началото на хендлера.

*на **излизане от хендлера** се изпълнява една последна специална инструкция, която кара μPU да копира обратно стековата група от SRAM в регистрите на ядрото. Това включва PC, в който се зарежда адреса, до който е било стигнало изпълнението на програмата преди прекъсването. Същото важи и за останалите регистри, които са специфични за всеки μPU (при MSP430 – SR, при ARM Cortex - PSR, R14, R12, R3, R2, R1, R0).

Обслужване на прекъсвания

Инструкцията за излизане от прекъсване при MSP430 е:

reti

Инструкцията за излизане от прекъсване при ARM Cortex е преход (branch) към адрес, записан в специален регистър (link register, lr = r14) :

b r14

Всъщност в r14 има служебен код, който казва на ядрото да възстанови стековата група и да направи преход към стойността на r14, която е PUSH-ната на стека.

Обслужване на прекъсвания

Исторически, цифровата схема, отговорна за съхраняване на векторите на прекъсване, приоритизирането им и максирането им, е била на отделен чип, наречен програмируем контролер на прекъсванията (PIC, **P**rogrammable **I**nterrupt **C**ontroller).

На схемата е показан класическият 8259A, съвместим с 8086, 8088.

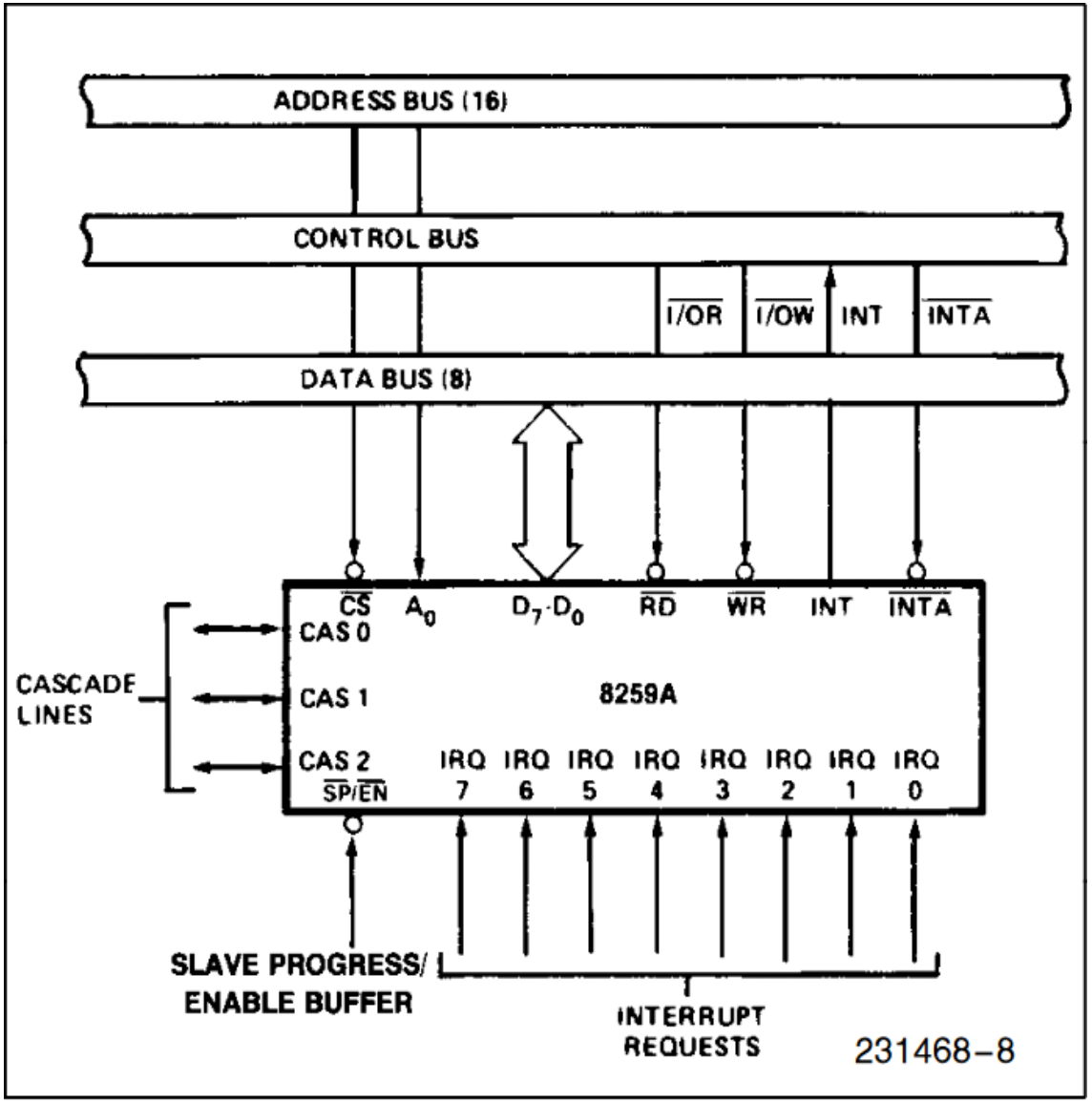


Figure 5. 8259A Interface to Standard System Bus

Обслужване на прекъсвания

В съвременните вградени системи **контролерът на прекъсванията е част от µPU** и се интегрира на чипа.

Източниците на прекъсвания се свързват към входовете на контролера на прекъсванията от **производителя**. Това означава, че приоритетите на прекъсванията са фиксирани от производителя.

ARM Cortex могат да **препрограмират приоритетите** по време на изпълнението на програмата.

При MSP430 всяка периферия има регистър, който указва точно от **кой източник е дошло прекъсването**, което ускорява обслужването му.

Обслужване на прекъсвания

Пример – MSP430, UART модул, IV регистър. Прилича на “малка векторна таблица”, локално в модула.

30.4.12 UCAxIV Register

eUSCI_Ax Interrupt Vector Register

Figure 30-23. UCAxIV Register

15	14	13	12	11	10	9	8
UCIVx							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
UCIVx							
r0	r0	r0	r0	r-(0)	r-(0)	r-(0)	r0

Table 30-19. UCAxIV Register Description

Bit	Field	Type	Reset	Description
15-0	UCIVx	R	0h	eUSCI_A interrupt vector value 00h = No interrupt pending 02h = Interrupt Source: Receive buffer full; Interrupt Flag: UCRXIFG; Interrupt Priority: Highest 04h = Interrupt Source: Transmit buffer empty; Interrupt Flag: UCTXIFG 06h = Interrupt Source: Start bit received; Interrupt Flag: UCSTTIFG 08h = Interrupt Source: Transmit complete; Interrupt Flag: UCTXCPTIFG; Interrupt Priority: Lowest

Обслужване на прекъсвания

```
void __attribute__ ((interrupt(USCI_A0_VECTOR))) USCI_A0_ISR (void){
  switch(UCA0IV)){
  case 0:
       //Vector 0 - no interrupt
      break;
  case 2:
       //Vector 2 - RXIFG
       ...
      break;
  case 4:
       //Vector 4 – TXIFG
       ...
      break;
  default:
      break;
  }
}
```

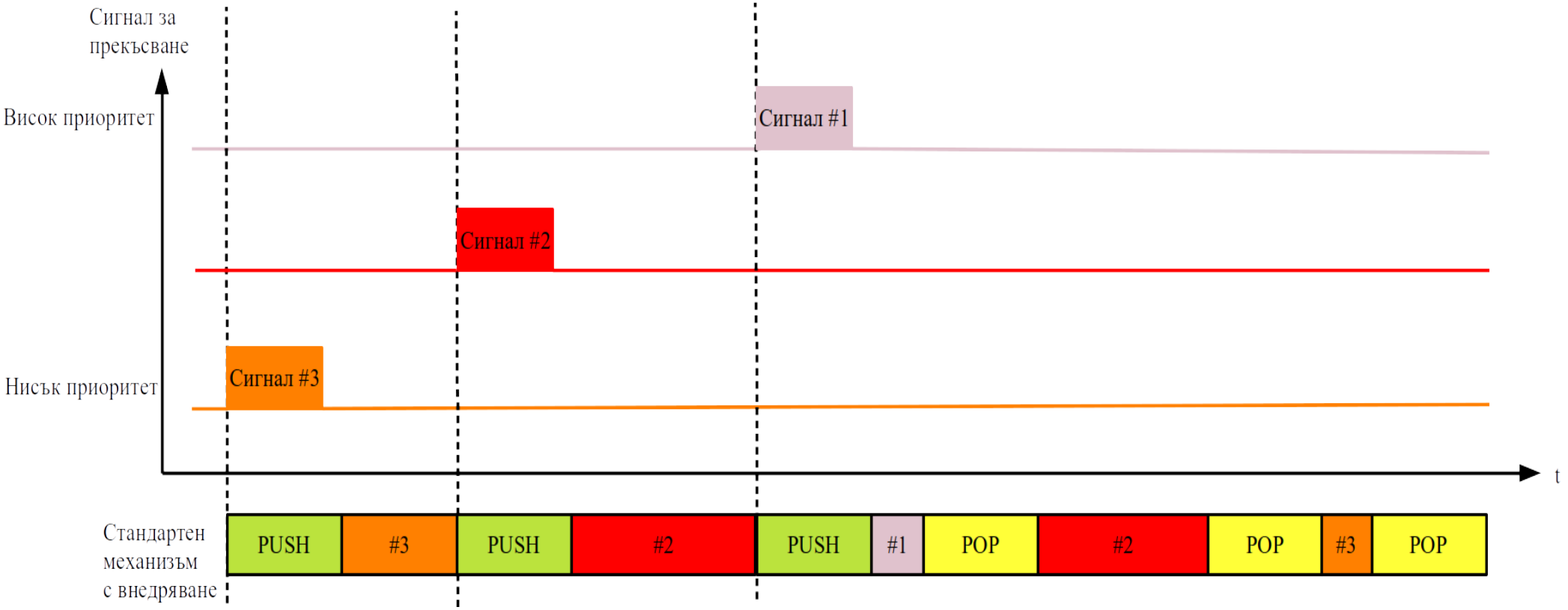
Обслужване на прекъсвания

Прекъсванията изискват специални механизми за обслужване, в случаите когато постъпят два или повече сигнала за прекъсване в един и същ момент.

Внедряване на прекъсванията (interrupt preemption) – изпълнението на хендлера на едно прекъсване може да бъде временно спряно, за да се изпълни хендлера на друго прекъсване. Това може да стане само, ако приоритета на новодошлото прекъсване е по-голям от настоящо-изпълняващото се прекъсване.

Не може да съществуват прекъсвания с един и същ приоритет (освен ако не се въведат суб-приоритети, както е при ARM Cortex).

Обслужване на прекъсвания

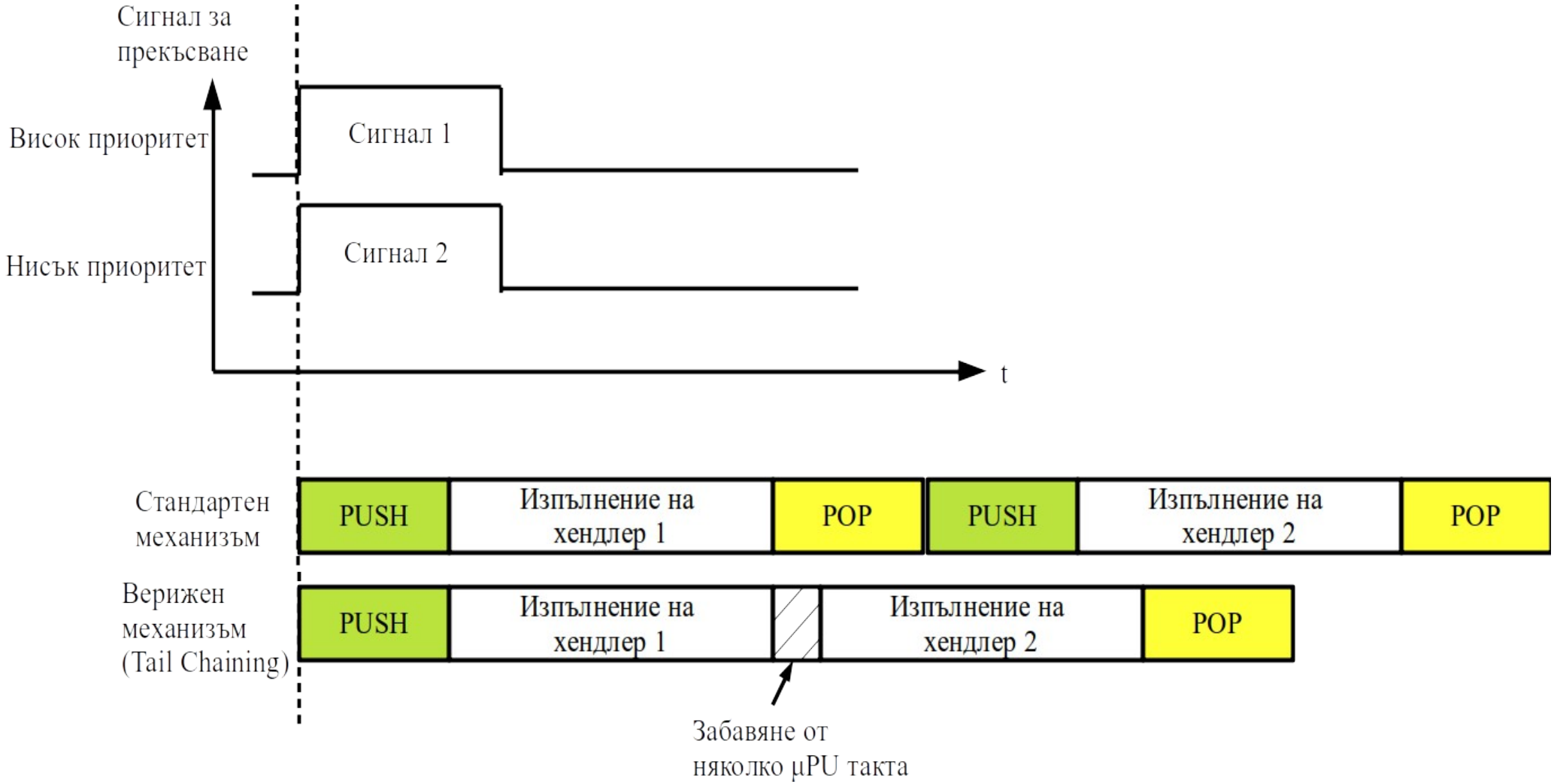


Обслужване на прекъсвания

Верижен механизъм (tail chaining) – при едновременно постъпване на сигнали за прекъсване, преминаването от хендлер 1 в хендлер 2 става без да се записва (PUSH) и възстановява (POP) стека. Резултатът – по-бързо обслужване на прекъсванията спрямо стандартния подход. Методът е демонстриран на следващия слайд[3].

Преминаването от един хендлер в друг не става мигновено, а изисква няколко системни такта, обусловени от вътрешната структура на μPU.

Обслужване на прекъсвания

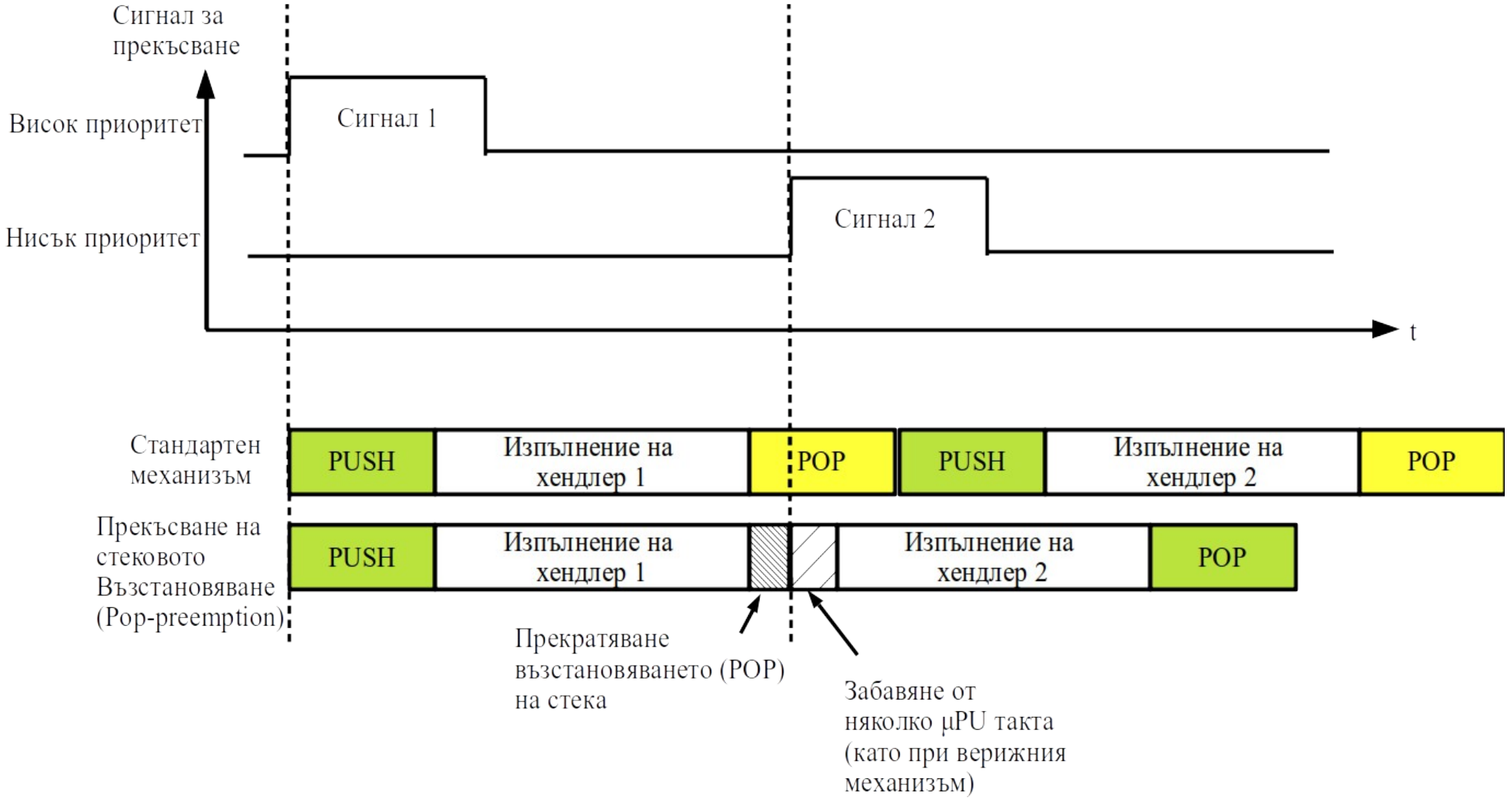


Обслужване на прекъсвания

Прекъсване на стековото възстановяване (pop pre-emption) – при постъпване на сигнал за прекъсване, докато μPU излиза от друго прекъсване, се прекратява възстановяването на стека (POP) и се преминава към изпълняване на следващия хендлер. Методът е демонстриран на следващия слайд.

Прекратяването на процеса по възстановяване на стека не може да стане мигновено и са необходими няколко системни такта за целите на микропроцесорната логика.

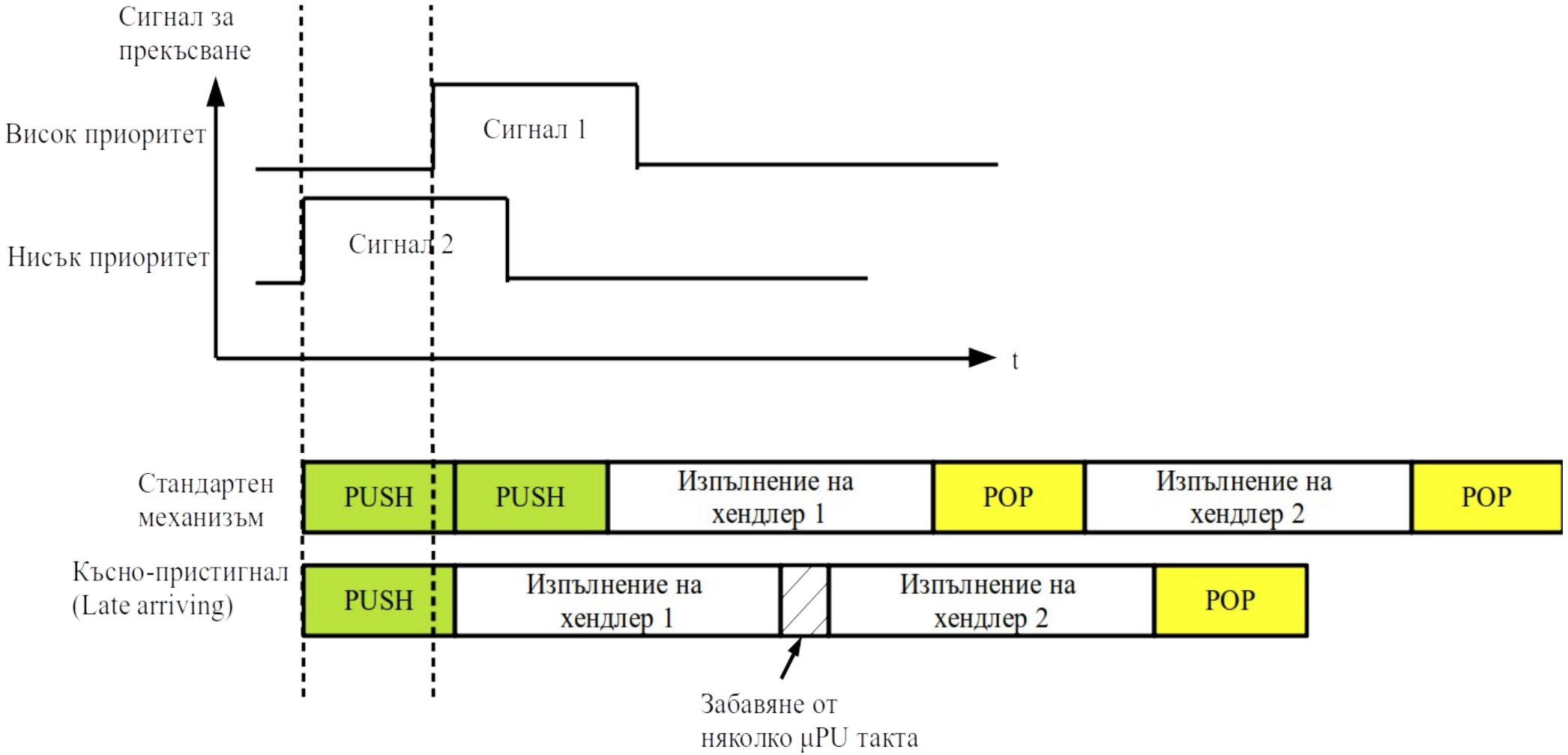
Обслужване на прекъсвания



Обслужване на прекъсвания

Късно-пристигнал (late arriving) – изпълнение на хендлер с висок приоритет във времевия слот на хендлер с нисък приоритет, ако първият е пристигнал по време на PUSH операцията на втория.

Обслужване на прекъсвания



Обслужване на прекъсвания

Флагове
Битове за разрешаване
Локални и глобални прекъсвания

Литература

[1]Pedro Dinis Gaspar, Antonio Santo, Bruno Ribeiro, Humberto Santos, Device Systems and Operating Modes, chapter 5, TI & University of Beira Interior (PT), 2009.

[2]Н. Кенаров, “PIC Микроконтролери”, Част 1, Млад Конструктор, Варна, 2003.

[3]M. Trevor, „The Designer’s Guide to the Cortex-M Processor Family – A Tutorial Approach“, Elsevier, 2013.