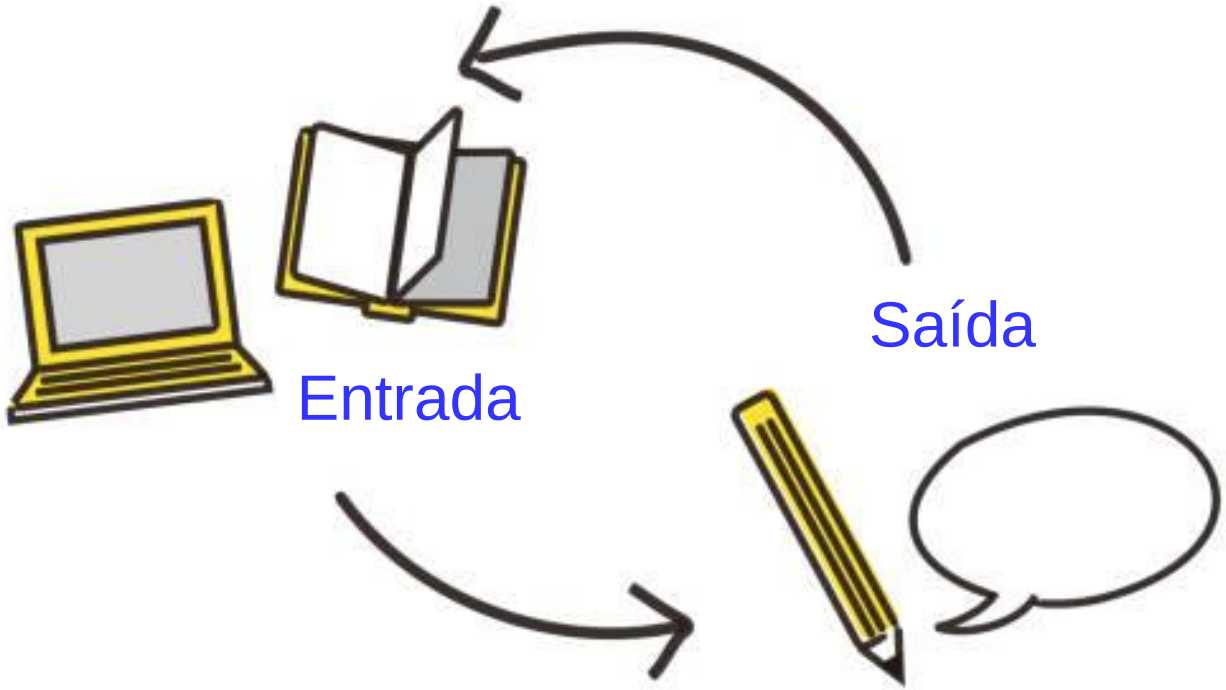
Organização de Computadores

Entrada e Saída

Prof. José Paulo G. de Oliveira
Eng. da Computação, UPE
jpgo@ecomp.poli.br

Interfaces de E/S



Interfaces digitais de E/S

- E/S não é só o conector físico
- É também o responsável pela comunicação lógica entre o barramento e o dispositivo
- Função de conexão que viabiliza a comunicação entre vários dispositivos
  - Velocidade do barramento mais bem aproveitada

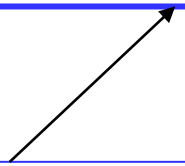
Interfaces digitais de E/S

- E/S não é só o conector físico

- É também o responsável pela comunicação lógica entre o barramento e o dispositivo

- Função de conexão que viabiliza a comunicação entre vários dispositivos

  – Velocidade do barramento mais bem aproveitada

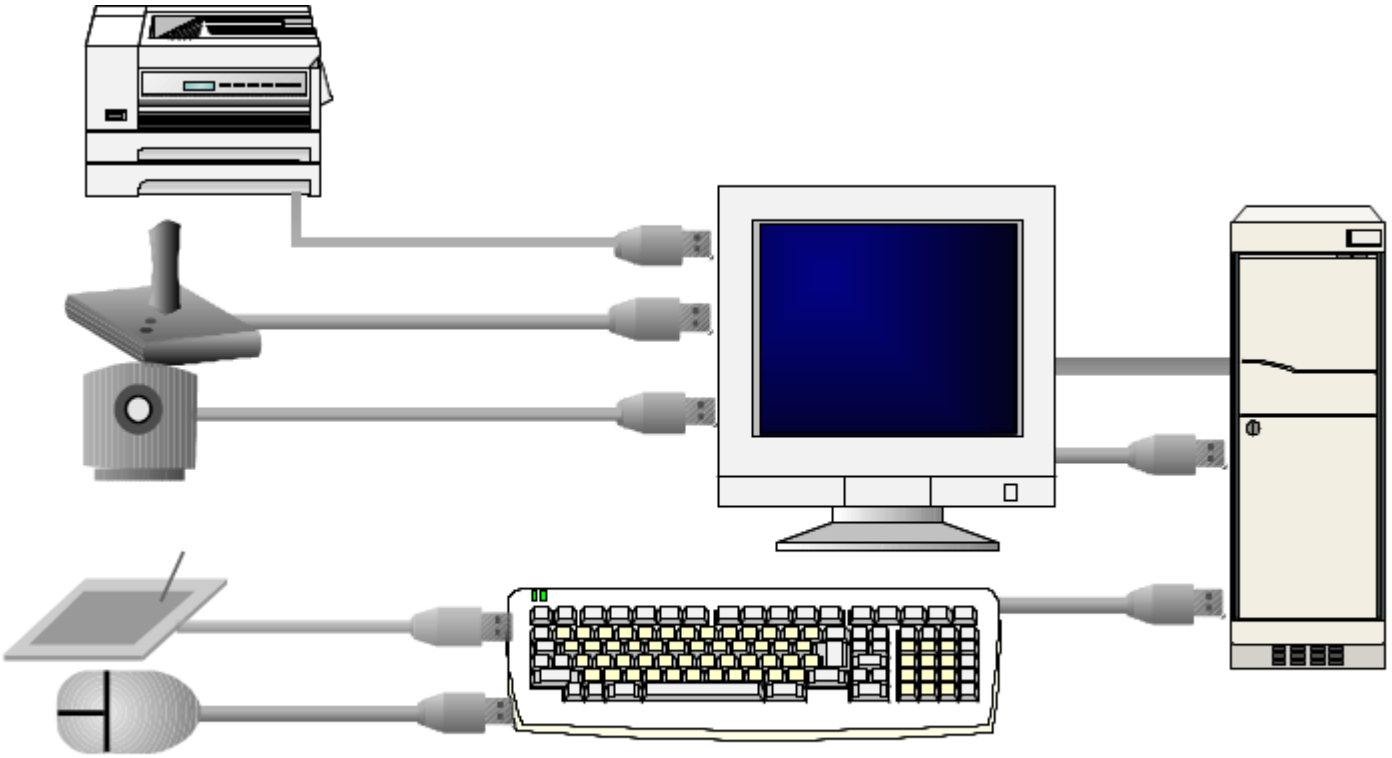


Escopo da disciplina

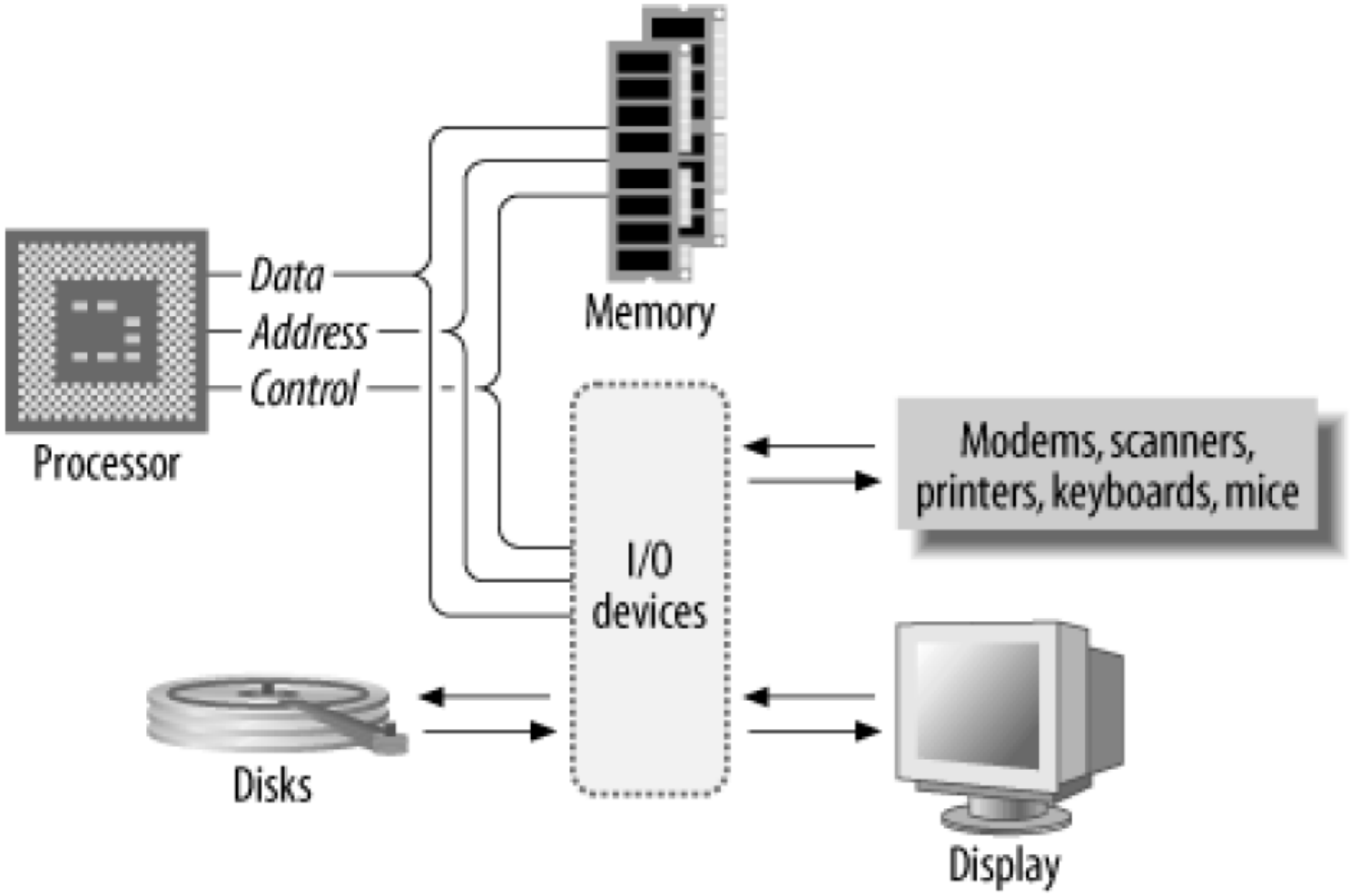
Resumo

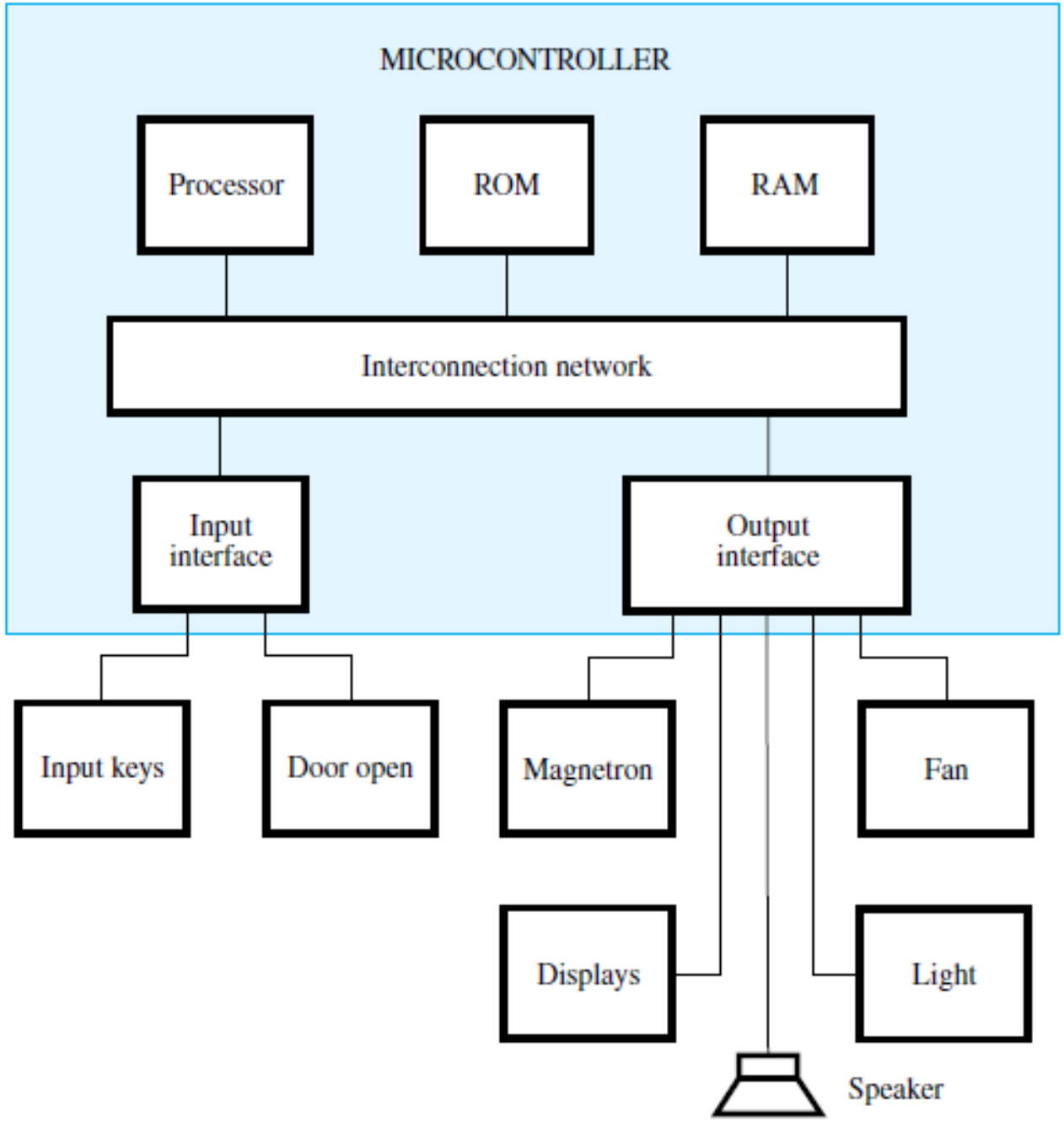
- Visão geral
- Dispositivos de E/S
- Módulos de E/S
- Técnicas de E/S
- Sistemas de alto desempenho

Exemplos de E/S - PC

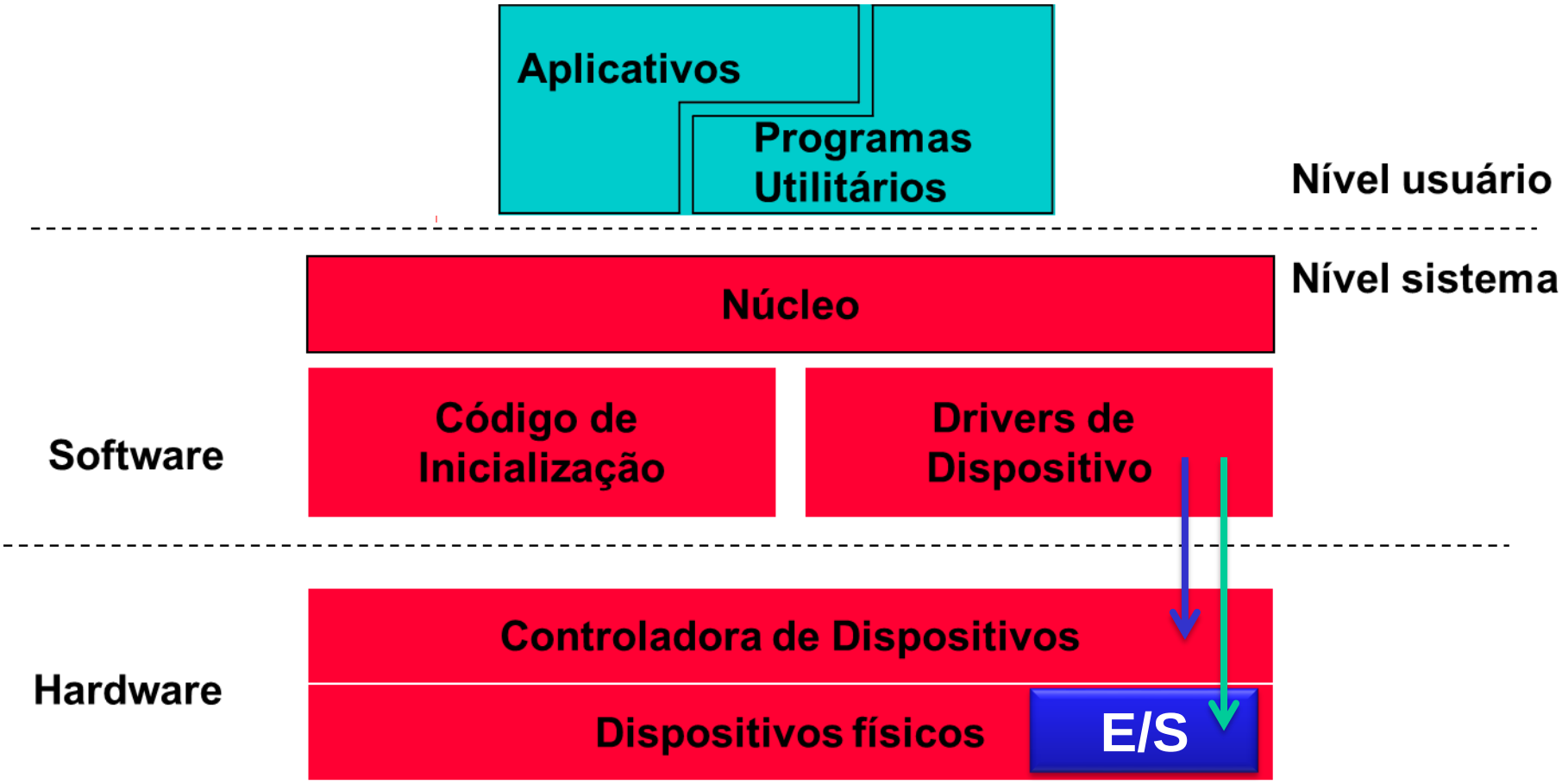


Estrutura de um Computador

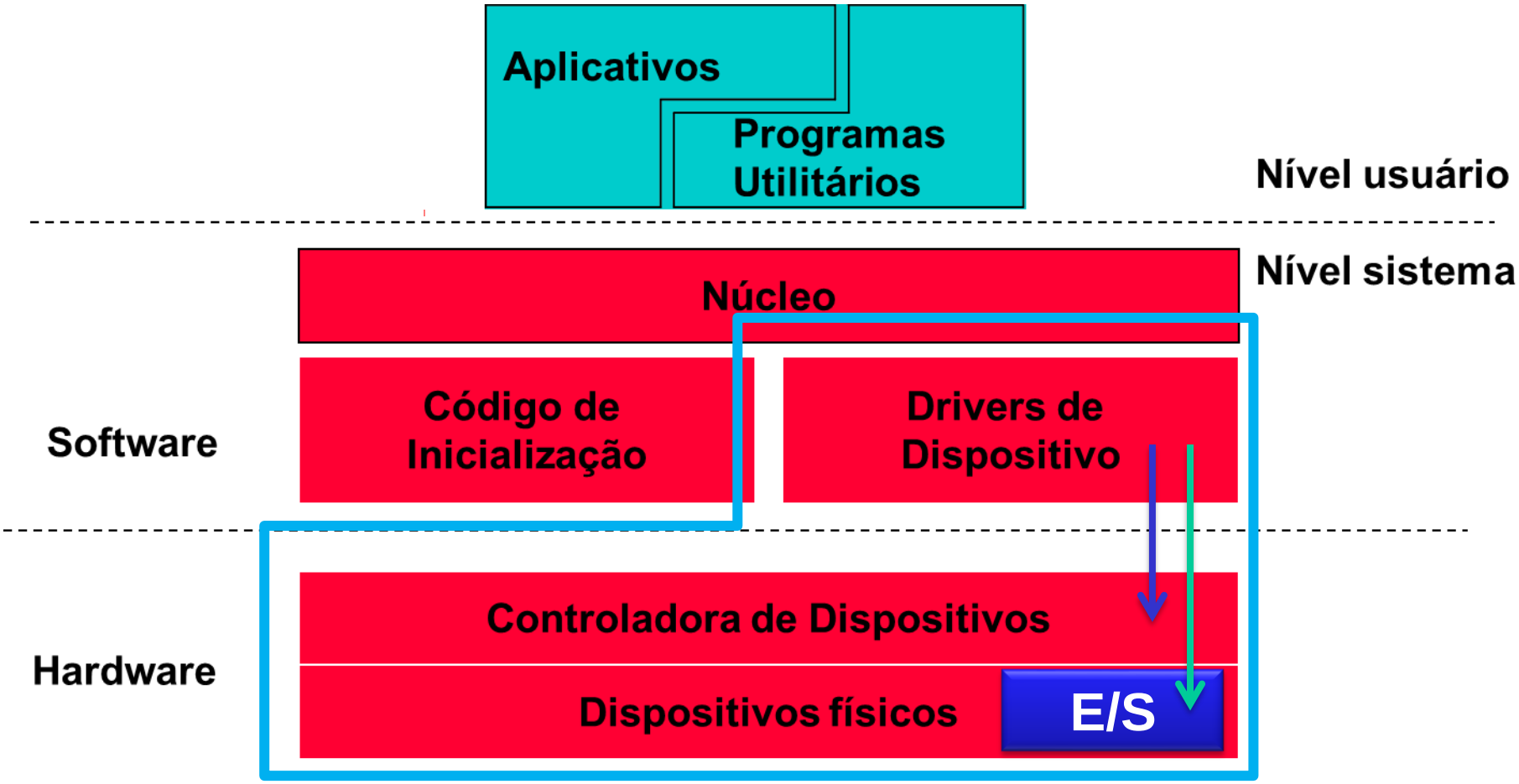



MICROCONTROLLER
Processor
ROM
RAM
Interconnection network
Input interface
Output interface
Input keys
Door open
Magnetron
Fan
Displays
Light
Speaker

Estrutura de um Computador



Estrutura de um Computador



Driver de E/S – Tiva C

25.2 API Functions

Functions

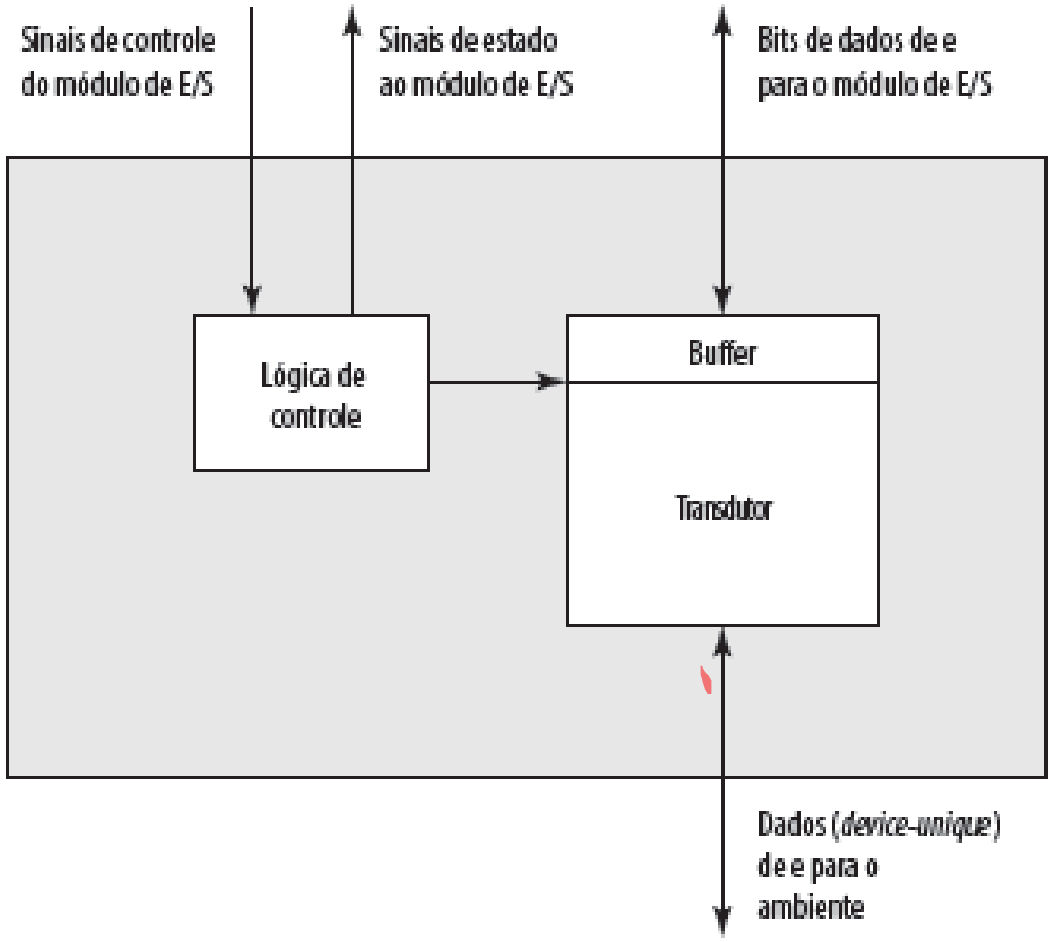
- void UART9BitAddrSend (uint32_t ui32Base, uint8_t ui8Addr)
- void UART9BitAddrSet (uint32_t ui32Base, uint8_t ui8Addr, uint8_t ui8Mask)
- void UART9BitDisable (uint32_t ui32Base)
- void UART9BitEnable (uint32_t ui32Base)
- void UARTBreakCtl (uint32_t ui32Base, bool bBreakState)
- bool UARTBusy (uint32_t ui32Base)

Dispositivos Externos - Operações

Operações de E/S são efetuadas por meio de uma grande variedade de dispositivos externos.

Dispositivo Externo – Modelo Geral

Interface com módulo de E/S →



Dispositivos Externos - Tipos

- Comunicação com Usuário
  - Tela, impressora, teclado
- Comunicação com a
  - Monitoramento e controle
  - Armazenamento (HDs, Drives ópticos)
- Comunicação com dispositivos remotos
  - Modem
  - Placas de rede

Dispositivos Externos - Tipos

- Periféricos específicos (sist. embarcados)
  - Sensores, atuadores
  - Conversores AD e DA
  - Interfaces de comunicação / Barramento
    - UART, CAN, I^2C, SPI

Dispositivos Externos - Velocidade

Dispositivo	velocidade
Teclado	10 B/s
Mouse ótico	100 B/s
Interface infravermelho (IrDA-SIR)	14 KB/s
Interface paralela padrão	125 KB/s
Interface de áudio digital S/PDIF	384 KB/s
Interface de rede *Fast Ethernet*	11.6 MB/s
Chave ou disco USB 2.0	60 MB/s
Interface de rede *Gigabit Ethernet*	116 MB/s
Disco rígido SATA 2	300 MB/s
Interface gráfica *high-end*	4.2 GB/s

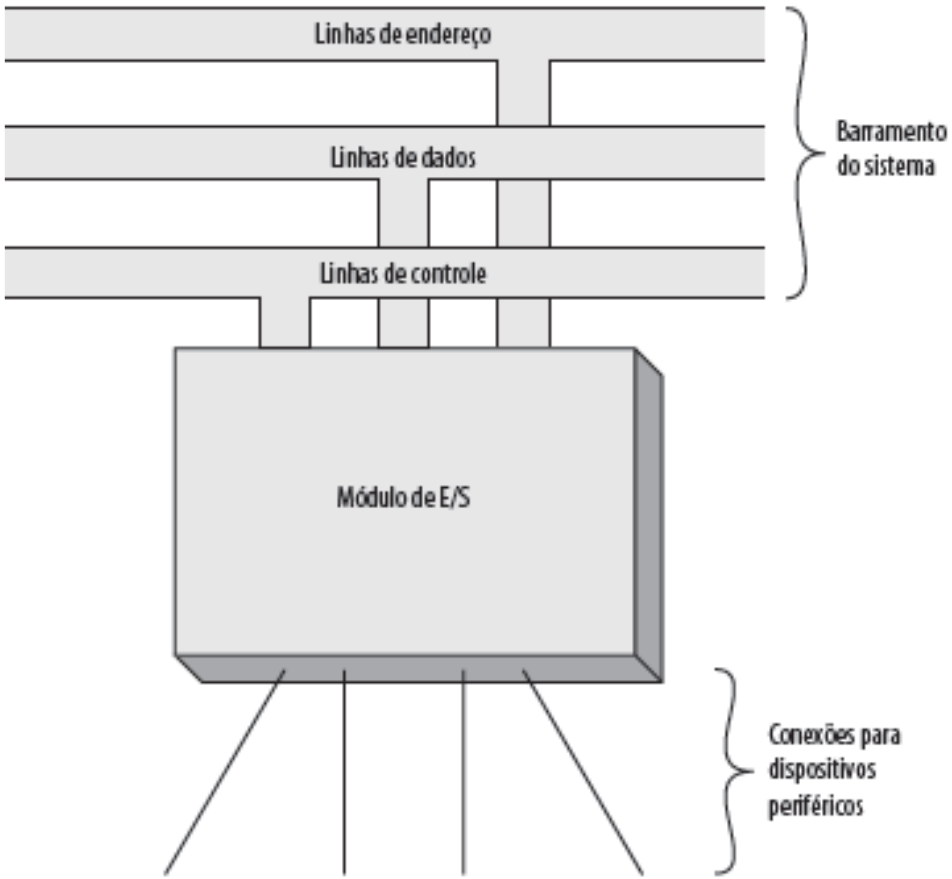
Problemas de Entrada e Saída

- Ampla variedade de periféricos
  - Diferentes tipos e quantidades de dados
  - Diferentes velocidades
  - Diferentes formatos
- Mais lento do que a CPU e a memória principal (em geral)

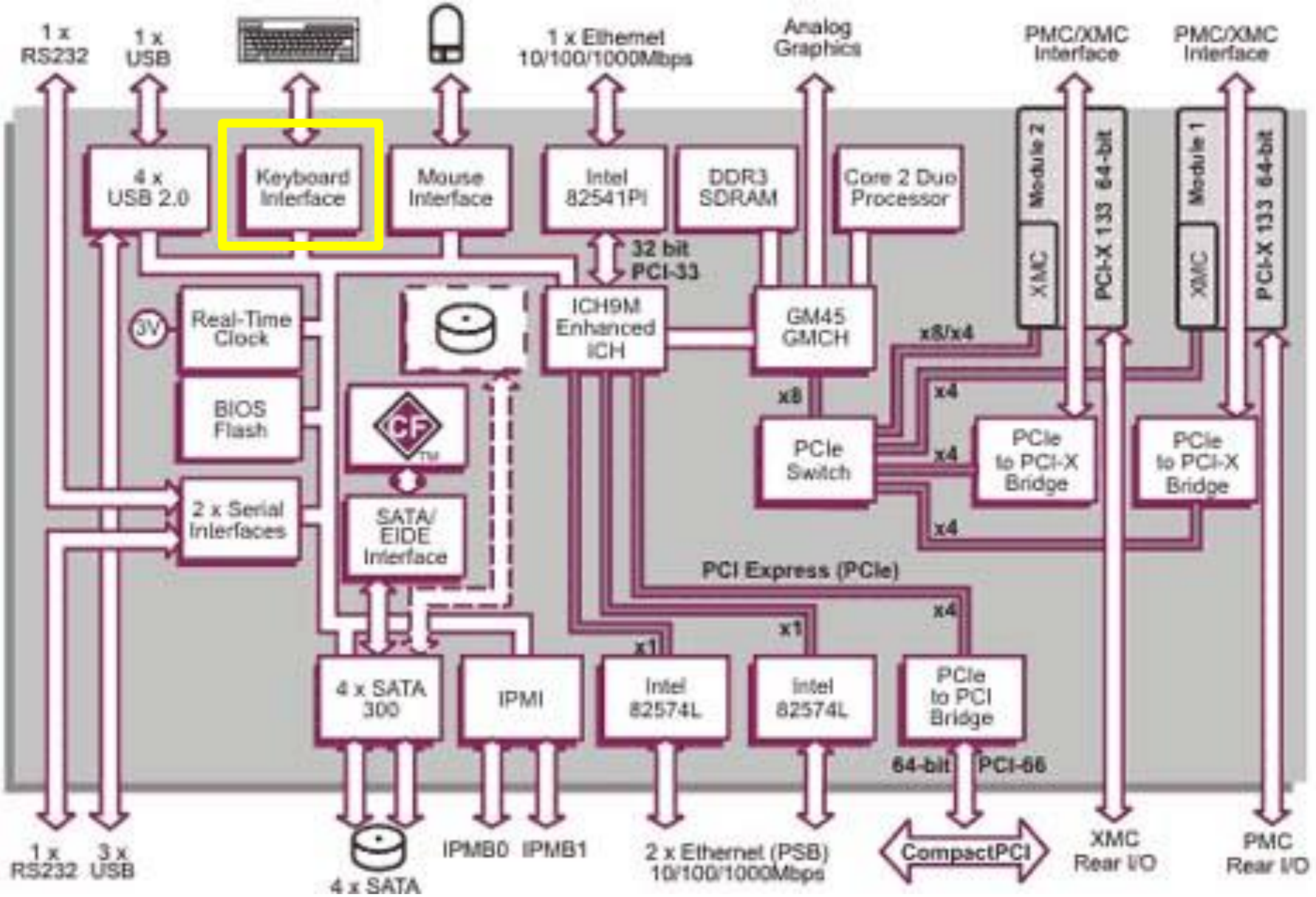
Problemas de Entrada e Saída

- Ampla variedade de periféricos
  - Diferentes tipos e quantidades de dados
  - Diferentes velocidades
  - Diferentes formatos
- Mais lento do que a CPU e a memória principal (em geral)
- Necessidade de módulos de E/S

Módulo de Entrada e Saída



- Interface para a CPU e a memória
- Interface para um ou mais periféricos


1 x RS232
1 x USB
1 x Ethernet 10/100/1000Mbps
Analog Graphics
PMC/XMC Interface
PMC/XMC Interface
4 x USB 2.0
Keyboard Interface
Mouse Interface
Intel 82541PI
DDR3 SDRAM
Core 2 Duo Processor
Module 2
XMC
PCI-X 133 64-bit
Module 1
XMC
PCI-X 133 64-bit
32 bit PCI-33
3V
Real-Time Clock
BIOS Flash
2 x Serial Interfaces
SATA/ EIDE Interface
ICH9M Enhanced ICH
GM45 GMCH
x8/x4
x8
x4
x4
x4
PCIe Switch
PCIe to PCI-X Bridge
PCIe to PCI-X Bridge
PCI Express (PCIe)
x4
x1
x1
4 x SATA 300
IPMI
Intel 82574L
Intel 82574L
PCIe to PCI Bridge
64-bit
PCI-66
1 x RS232
3 x USB
4 x SATA
IPMB0
IPMB1
2 x Ethernet (PSB) 10/100/1000Mbps
CompactPCI
XMC Rear I/O
PMC Rear I/O

Função e Estrutura do Módulo de E/S

Função do Módulo de E/S

- Controle e temporização
- Comunicação com a CPU
- Comunicação com dispositivos
- Armazenamento temporário de dados (*data buffering*)
- Detecção de erros

Função do Módulo de E/S

- Controle e temporização
- Comunicação com a CPU
- Comunicação com dispositivos
- Armazenamento temporário de dados (*data buffering*)
- Detecção de erros

Função do Módulo de E/S

A CPU pode se comunicar com um ou mais dispositivos durante a execução de um programa.

→ Recursos internos compartilhados

→ Gerenciamento: controle e temporização

Função do Módulo de E/S

- Controle e temporização
- Comunicação com a CPU
- Comunicação com dispositivos
- Armazenamento temporário de dados (*data buffering*)
- Detecção de erros

Ex.: Passos da operação de E/S

1. A CPU interroga sobre o status de dispositivo do módulo de E/S
2. O módulo de E/S retorna o status
3. Se o módulo está pronto, a CPU requisita transferência de dados
4. O módulo de E/S adquire dados do dispositivo
5. O módulo de E/S transfere dados para a CPU

Função do Módulo de E/S

- Controle e temporização
- Comunicação com a CPU
- Comunicação com dispositivos
- Armazenamento temporário de dados (*data buffering*)
- Detecção de erros

Armazenamento de dados (*buffering*)

Dispositivo	velocidade
Teclado	10 B/s
Mouse ótico	100 B/s
Interface infravermelho (IrDA-SIR)	14 KB/s
Interface paralela padrão	125 KB/s
Interface de áudio digital S/PDIF	384 KB/s
Interface de rede *Fast Ethernet*	11.6 MB/s
Chave ou disco USB 2.0	60 MB/s
Interface de rede *Gigabit Ethernet*	116 MB/s
Disco rígido SATA 2	300 MB/s
Interface gráfica *high-end*	4.2 GB/s

Taxas **muito menores** que as alcançadas nas transferências entre **CPU e memória**

Armazenamento de dados (*buffering*)

Dispositivo	velocidade
Teclado	10 B/s
Mouse ótico	100 B/s
Interface infravermelho (IrDA-SIR)	14 KB/s
Interface paralela padrão	125 KB/s
Interface de áudio digital S/PDIF	384 KB/s
Interface de rede *Fast Ethernet*	11.6 MB/s
Chave ou disco USB 2.0	60 MB/s
Interface de rede *Gigabit Ethernet*	116 MB/s
Disco rígido SATA 2	300 MB/s
Interface gráfica *high-end*	4.2 GB/s

Além disso, os canais de comunicação do sistema computacional podem estar **ocupados**.

Função do Módulo de E/S

- Controle e temporização
- Comunicação com a CPU
- Comunicação com dispositivos
- Armazenamento temporário de dados (*data buffering*)
- Detecção de erros

Detecção de erros

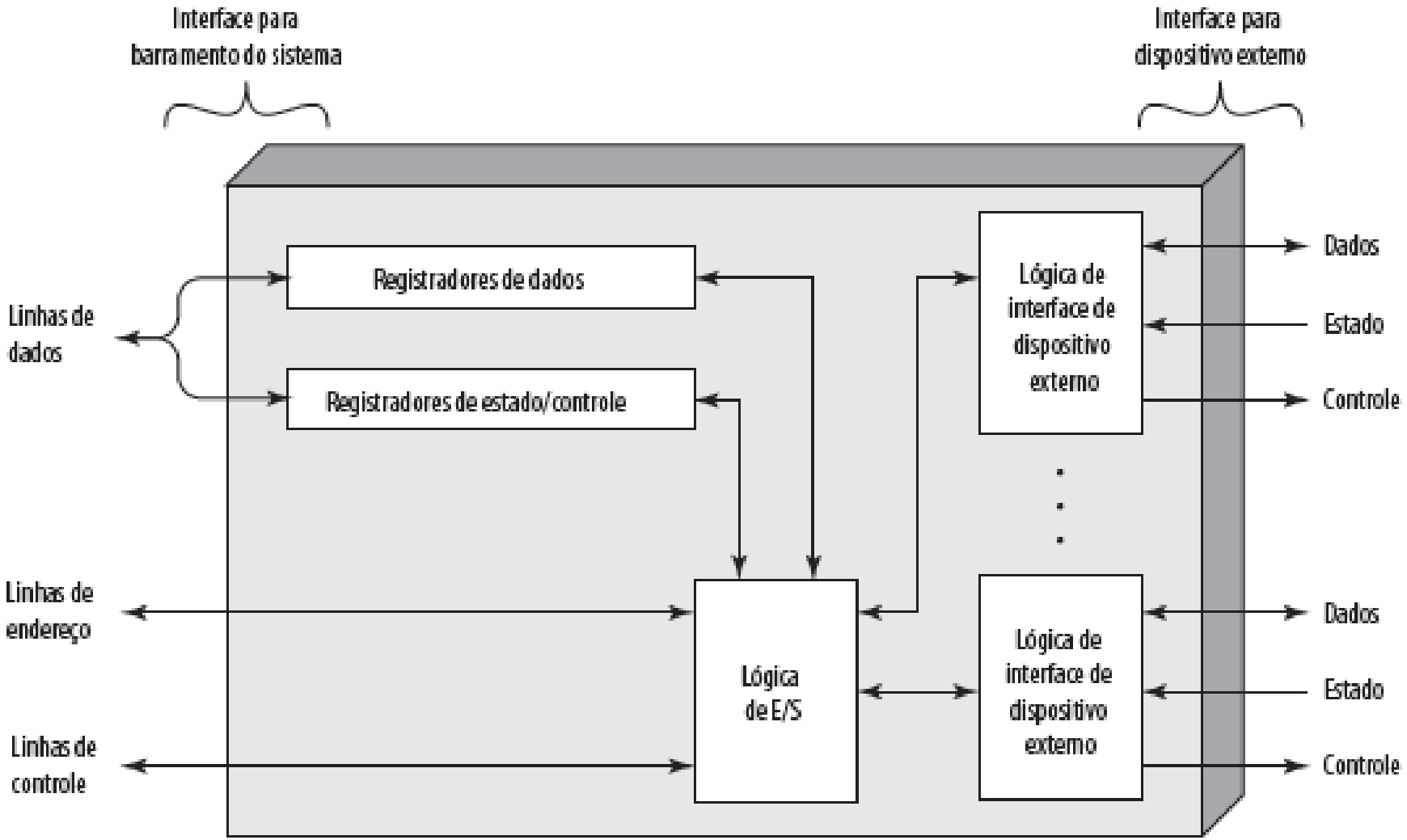
- Erros de operação
  - Ex.: Falta de papel; Falha no canal de comunicação
- Erros de transmissão de dados
  - Uso de códigos detectores/corretores de erro

Estrutura de um Módulo de E/S

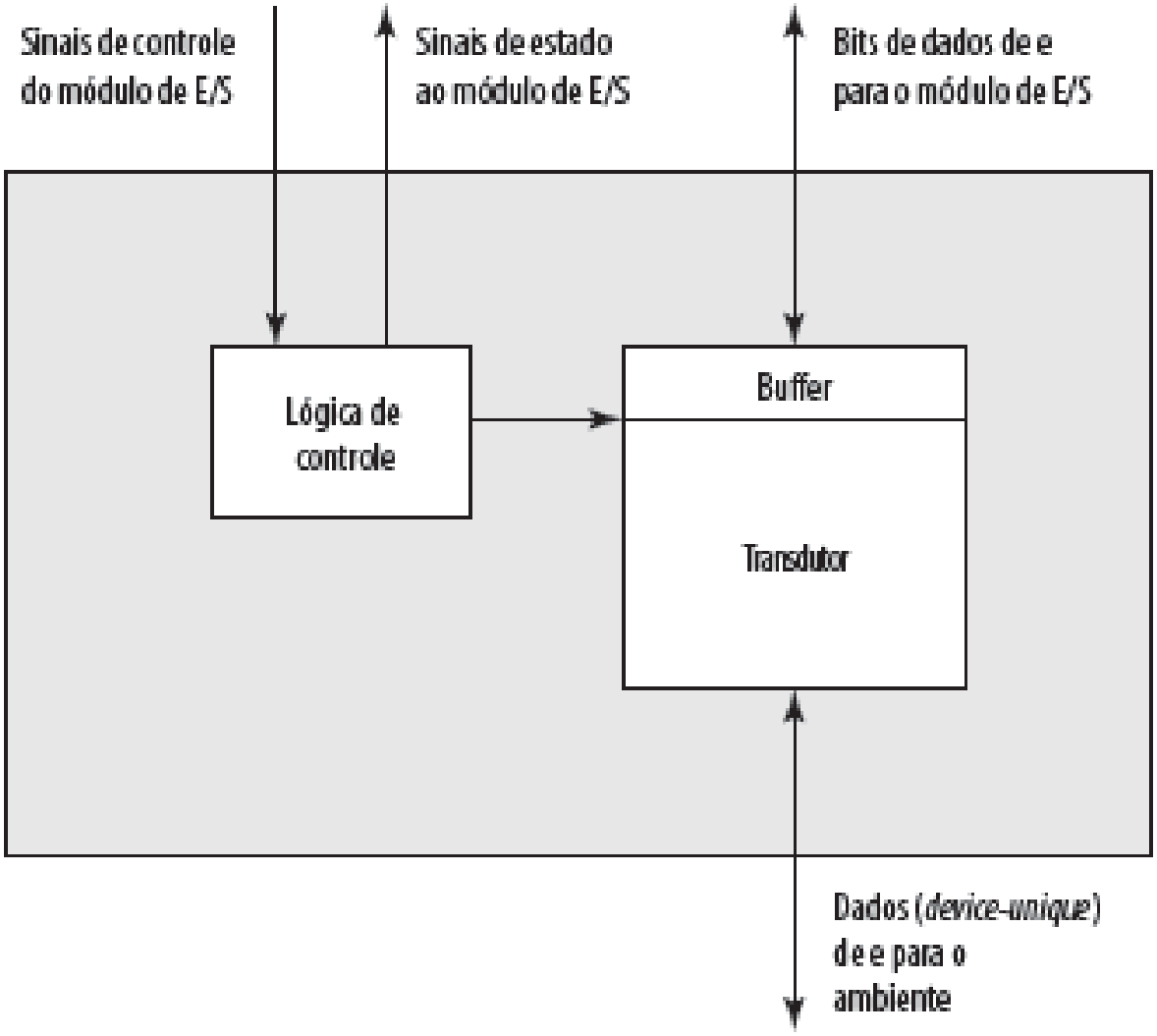
A complexidade de um módulo de E/S e o número de dispositivos externos que ele controla variam muito.

→ Necessidade de um “modelo genérico”

Estrutura de um Módulo de E/S



Relembrando: Dispositivo Externo



Objetivos de um Módulo de E/S

1. Esconder as propriedades do dispositivo para a CPU
2. Simplificar o suporte a múltiplos dispositivo

Objetivos de um Módulo de E/S

3. Controlar as funções do dispositivo e liberar a CPU dessa tarefa
   - Controladores, Processadores e canais de E/S
5. Implementar operações de SO
   - Ex.: sistemas baseados em Unix tratam dispositivos como arquivo

Técnicas de E/S

- Programada
- Executada por Interrupção
- Direct Memory Access (DMA)

Técnicas de E/S

- Programada
- Executada por Interrupção
- Direct Memory Access (DMA)

E/S Programada

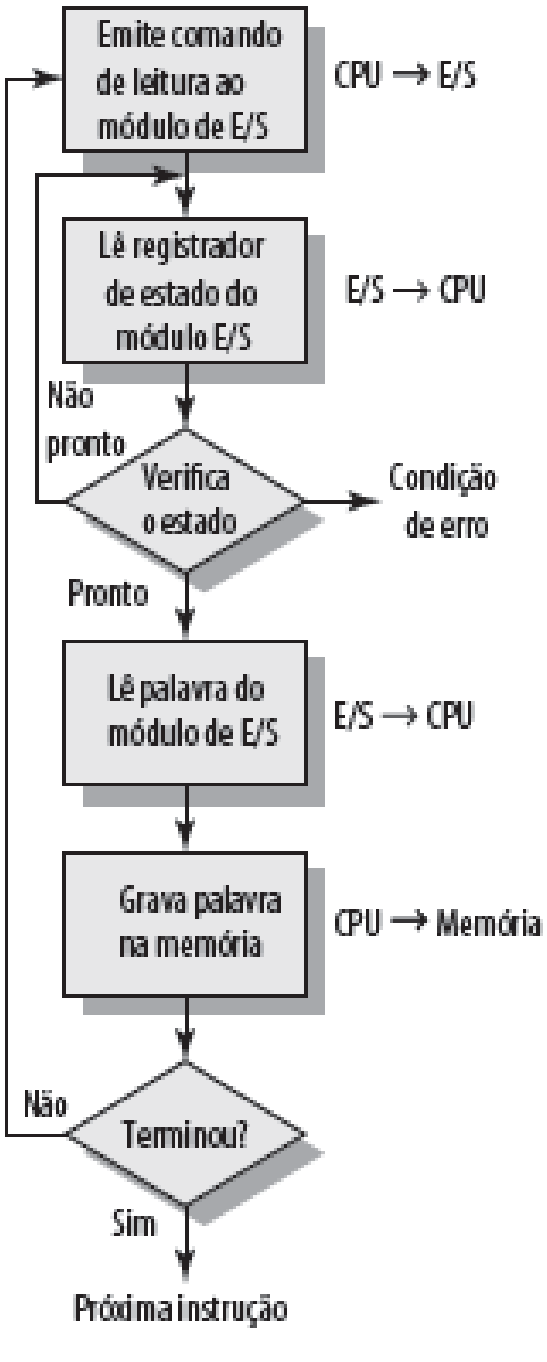
- A CPU tem controle direto sobre a E/S
  - Comandos de leitura e escrita
  - Monitoramento constante do status
  - Transferência de dados
- A CPU espera até que o módulo de E/S complete a operação

E/S Programada

- A CPU tem controle direto sobre a E/S
  - Comandos de leitura e escrita
  - Monitoramento constante do status
  - Transferência de dados
- A CPU espera até que o módulo de E/S complete a operação

$\Rightarrow$ Desperdício do tempo da CPU

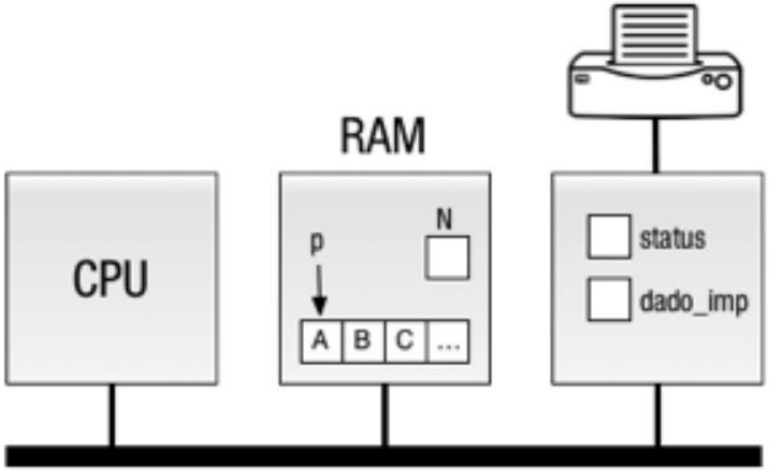
E/S Programada: Detalhamento



E/S Programada: Detalhamento

1. A CPU requisita operação da E/S
2. O módulo de E/S inicia a operação
3. O módulo de E/S atualiza os respectivos bits de status
4. A CPU verifica o status dos bits periodicamente
5. O módulo de E/S não informa a CPU prontamente
   - O módulo de E/S não interrompe a CPU
6. A CPU pode esperar ou voltar mais tarde (*Polling*)

E/S Programada: Ex.: impressão




```
copia(dados,udados,N); /* copia para espaço kernel */
for (cont=0; cont<N; ++cont) {
    while (*status != PRONTO) {/* espera ocupada */}
    *dado_imp = p;
    ++p;
}
```

Exemplo ADC Tiva C

```c
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/debug.h"
#include "driverlib/sysctl.h"
#include "driverlib/adc.h"

int main(void)
{
    uint32_t ui32ADC0Value[4];
    volatile uint32_t ui32TempAvg;
    volatile uint32_t ui32TempValueC;
    volatile uint32_t ui32TempValueF;

    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);

    ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE,1,3,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
    ADCSequenceEnable(ADC0_BASE, 1);

    while(1)
    {
       ADCIntClear(ADC0_BASE, 1);
       ADCProcessorTrigger(ADC0_BASE, 1);

       while(!ADCIntStatus(ADC0_BASE, 1, false))
       {
       }

       ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
       ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;
       ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
       ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;
    }
}
```

Comandos e Endereço de E/S

Para executar uma operação de E/S a CPU gera um comando de E/S + um endereço.

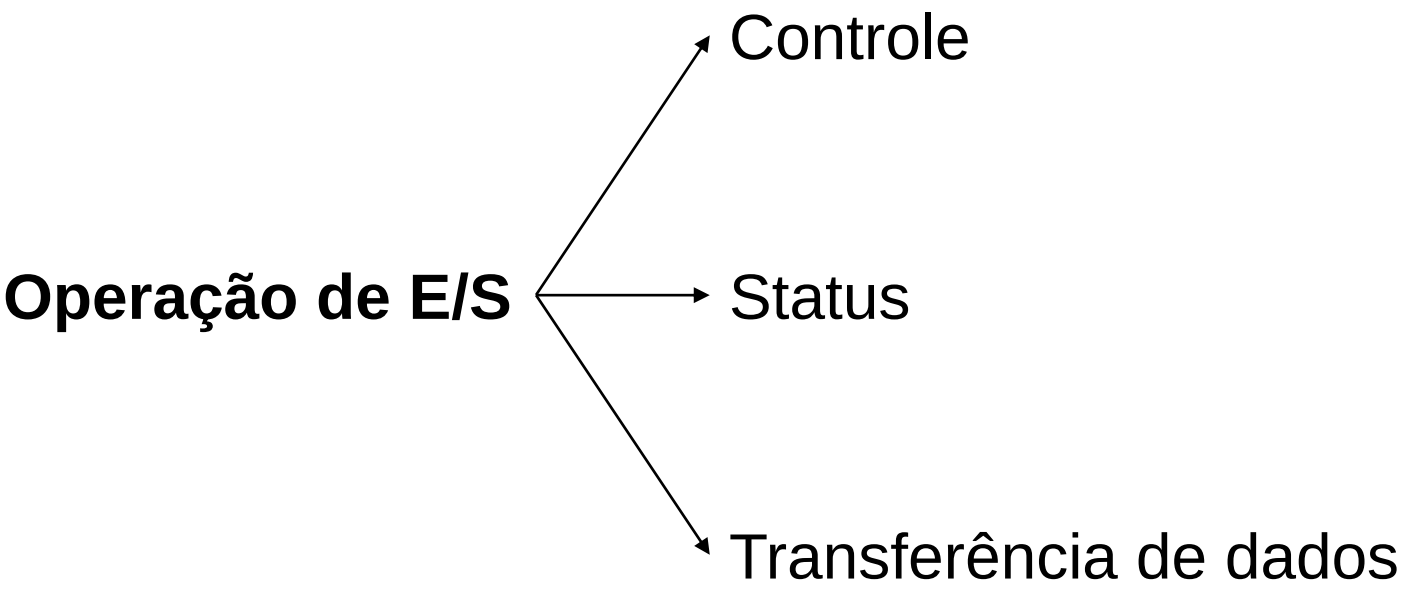
Comandos e Endereço de E/S

- CPU: endereço
  - Identifica o módulo (e também o dispositivo, se houver mais de um por módulo)

Comandos e Endereço de E/S

- CPU: endereço
  - Identifica o módulo (e também o dispositivo, se houver mais de um por módulo)
- CPU: comandos
  - Controle: diz ao módulo o que fazer
    - ex.: gire o disco
  - Teste: verifica o status
    - ex.: Está alimentado? Ocorreram erros?
  - Leitura e escrita
    - O módulo transfere dados do dispositivo ou para o dispositivo usando um buffer

Comandos e Endereço de E/S



Endereçamento de Dispositivos de E/S

- A transferência de dados é muito similar ao acesso à memória (do ponto de vista da CPU)
  - Cada dispositivo tem um único identificador
  - Os comandos da CPU contêm o identificador (endereço)
  - Cada módulo de E/S detecta se o comando é para ele ou não

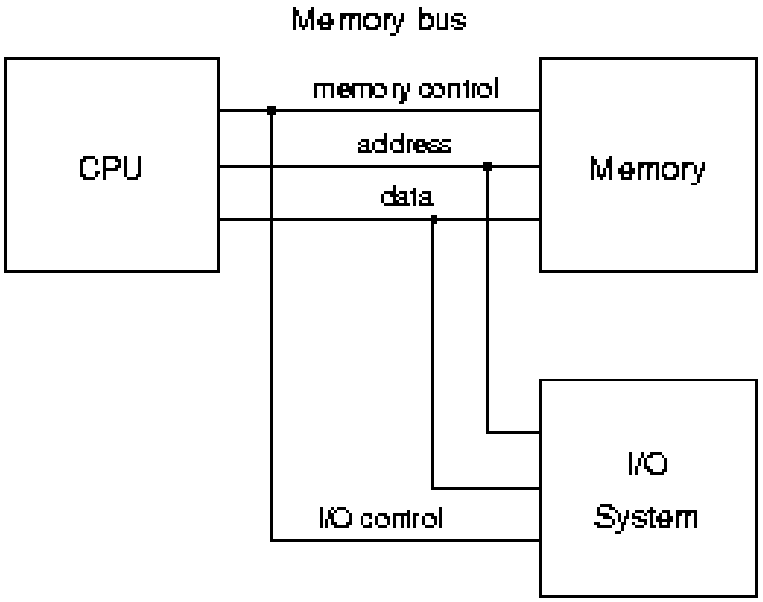
Endereçamento de Dispositivos de E/S

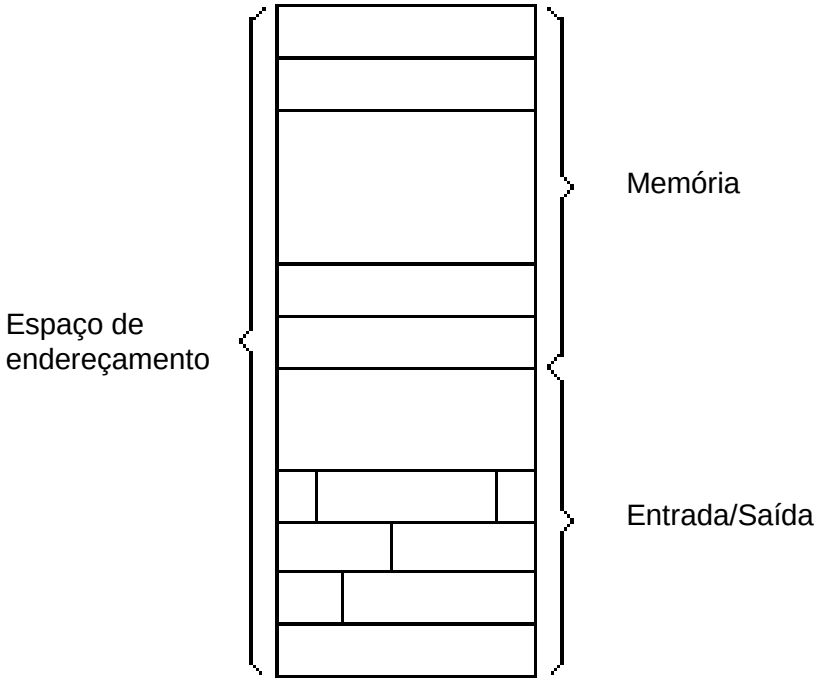
1. E/S mapeada em memória
2. E/S mapeada de forma independente

Mapeamento de E/S

- E/S mapeada em Memória
  - Dispositivos e memória compartilham uma faixa de endereços
    - Único espaço de endereçamento
  - E/S se assemelha a escrita e leitura em memória
  - Não há comandos específicos para E/S
    - Definição por endereço

Mapeamento de E/S

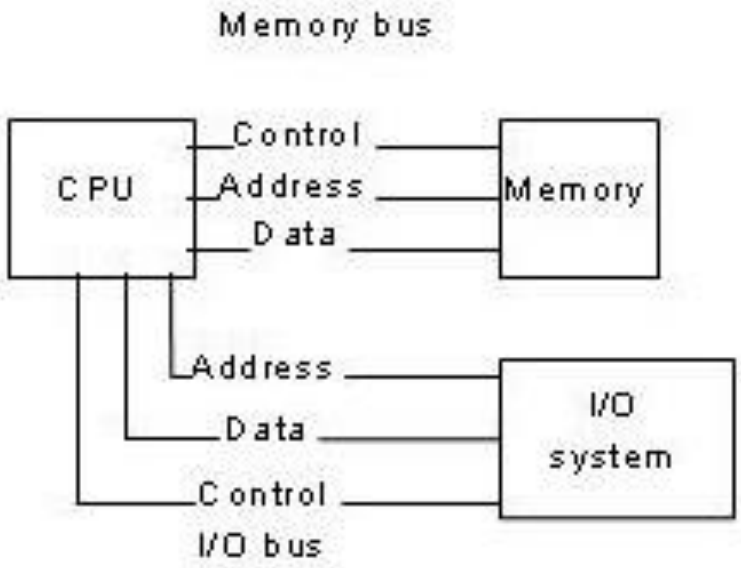


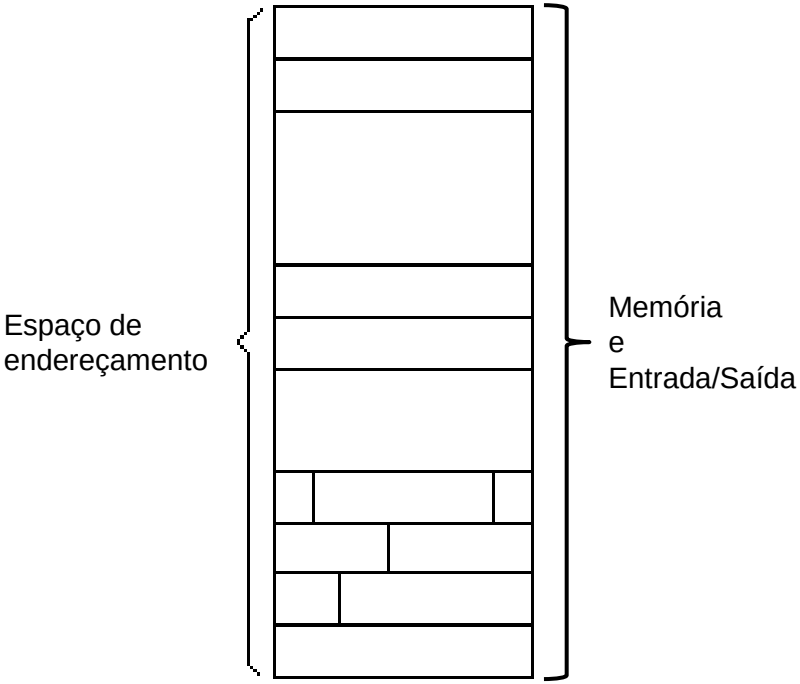


Mapeamento de E/S

- E/S mapeada em Memória
  - Dispositivos e memória compartilham uma faixa de endereços
    - Único espaço de endereçamento
  - E/S se assemelha a escrita e leitura em memória
  - Não há comandos específicos para E/S
    - Definição por endereço
- E/S independente
  - Espaços de endereço separados
  - Necessita de linhas de seleção de E/S ou memória
  - Comandos especiais para E/S

Mapeamento de E/S





Mapeamento - LM4F120H5QR

Address Range	Memory Region	Memory Type	Execute Never (XN)	Description
0x0000.0000 - 0x1FFF.FFFF	Code	Normal	-	This executable region is for program code. Data can also be stored here.
0x2000.0000 - 0x3FFF.FFFF	SRAM	Normal	-	This executable region is for data. Code can also be stored here. This region includes bit band and bit band alias areas (see Table 2-6 on page 95).
0x4000.0000 - 0x5FFF.FFFF	Peripheral	Device	XN	This region includes bit band and bit band alias areas (see Table 2-7 on page 96).
0x6000.0000 - 0x9FFF.FFFF	External RAM	Normal	-	This executable region is for data.

Address Range	Memory Region	Memory Type	Execute Never (XN)	Description
0xA000.0000 - 0xDFFF.FFFF	External device	Device	XN	This region is for external device memory.
0xE000.0000- 0xE00F.FFFF	Private peripheral bus	Strongly Ordered	XN	This region includes the NVIC, system timer, and system control block.
0xE010.0000- 0xFFFF.FFFF	Reserved	-	-	-

Mapeamento - LM4F120H5QR

Start	End	Description
0x4000.7000	0x4000.7FFF	GPIO Port D
0x4000.8000	0x4000.8FFF	SSI0
0x4000.9000	0x4000.9FFF	SSI1
0x4000.A000	0x4000.AFFF	SSI2
0x4000.B000	0x4000.BFFF	SSI3
0x4000.C000	0x4000.CFFF	UART0
0x4000.D000	0x4000.DFFF	UART1
0x4000.E000	0x4000.EFFF	UART2
0x4000.F000	0x4000.FFFF	UART3
0x4001.0000	0x4001.0FFF	UART4
0x4001.1000	0x4001.1FFF	UART5
0x4001.2000	0x4001.2FFF	UART6
0x4001.3000	0x4001.3FFF	UART7
0x4001.4000	0x4001.FFFF	Reserved

Mapeamento - LM4F120H5QR

Start	End	Description
0x4000.7000	0x4000.7FFF	GPIO Port D
0x4000.8000	0x4000.8FFF	SSI0
0x4000.9000	0x4000.9FFF	SSI1
0x4000.A000	0x4000.AFFF	SSI2
0x4000.B000	0x4000.BFFF	SSI3
0x4000.C000	0x4000.CFFF	UART0
0x4000.D000	0x4000.DFFF	UART1
0x4000.E000	0x4000.EFFF	UART2
0x4000.F000	0x4000.FFFF	UART3
0x4001.0000	0x4001.0FFF	UART4
0x4001.1000	0x4001.1FFF	UART5
0x4001.2000	0x4001.2FFF	UART6
0x4001.3000	0x4001.3FFF	UART7
0x4001.4000	0x4001.FFFF	Reserved

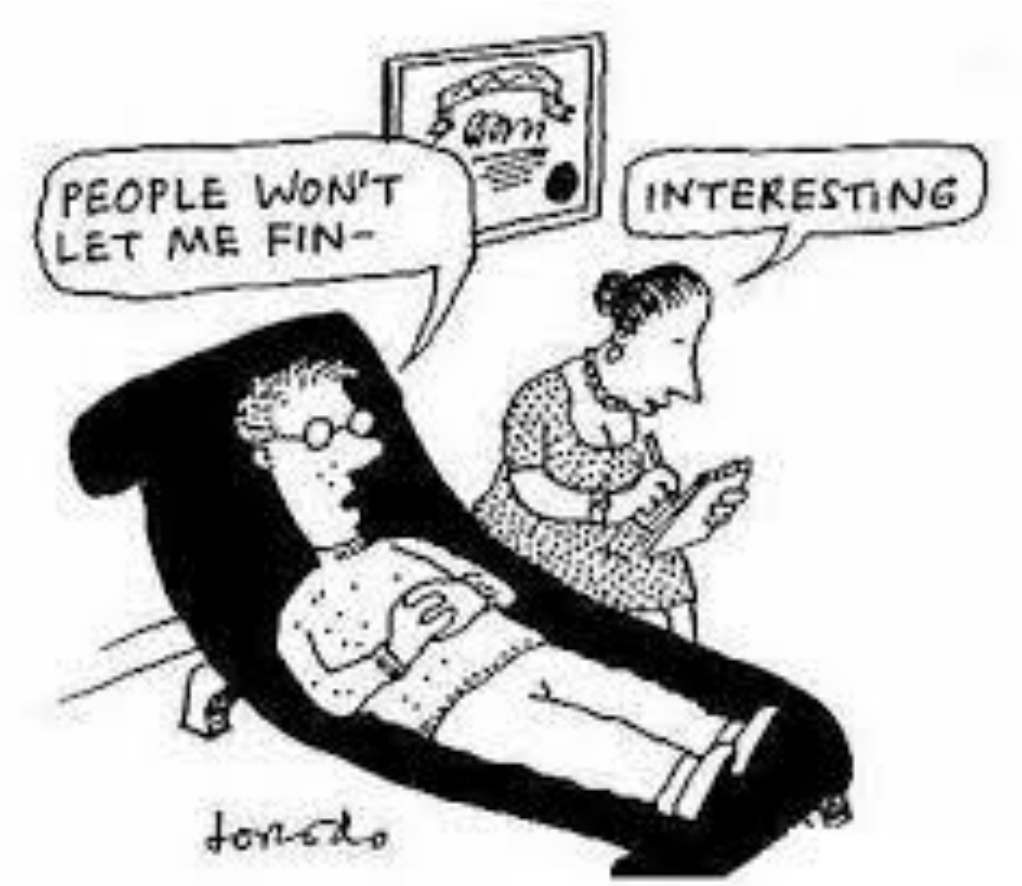
4096 posições

Start	End	Description
Peripherals		
0x4002.0000	0x4002.0FFF	I^2C 0
0x4002.1000	0x4002.1FFF	I^2C 1
0x4002.2000	0x4002.2FFF	I^2C 2
0x4002.3000	0x4002.3FFF	I^2C 3
0x4002.4000	0x4002.4FFF	GPIO Port E
0x4002.5000	0x4002.5FFF	GPIO Port F
0x4002.6000	0x4002.FFFF	Reserved
0x4003.0000	0x4003.0FFF	16/32-bit Timer 0
0x4003.1000	0x4003.1FFF	16/32-bit Timer 1
0x4003.2000	0x4003.2FFF	16/32-bit Timer 2
0x4003.3000	0x4003.3FFF	16/32-bit Timer 3
0x4003.4000	0x4003.4FFF	16/32-bit Timer 4
0x4003.5000	0x4003.5FFF	16/32-bit Timer 5
0x4003.6000	0x4003.6FFF	32/64-bit Timer 0
0x4003.7000	0x4003.7FFF	32/64-bit Timer 1
0x4003.8000	0x4003.8FFF	ADC0
0x4003.9000	0x4003.9FFF	ADC1
0x4003.A000	0x4003.BFFF	Reserved
0x4003.C000	0x4003.CFFF	Analog Comparators
0x4003.D000	0x4003.FFFF	Reserved
0x4004.0000	0x4004.0FFF	CAN0 Controller
0x4004.1000	0x4004.BFFF	Reserved
0x4004.C000	0x4004.CFFF	32/64-bit Timer 2

Técnicas de E/S

- Programada
- Comandada por Interrupção
- Direct Memory Access (DMA)

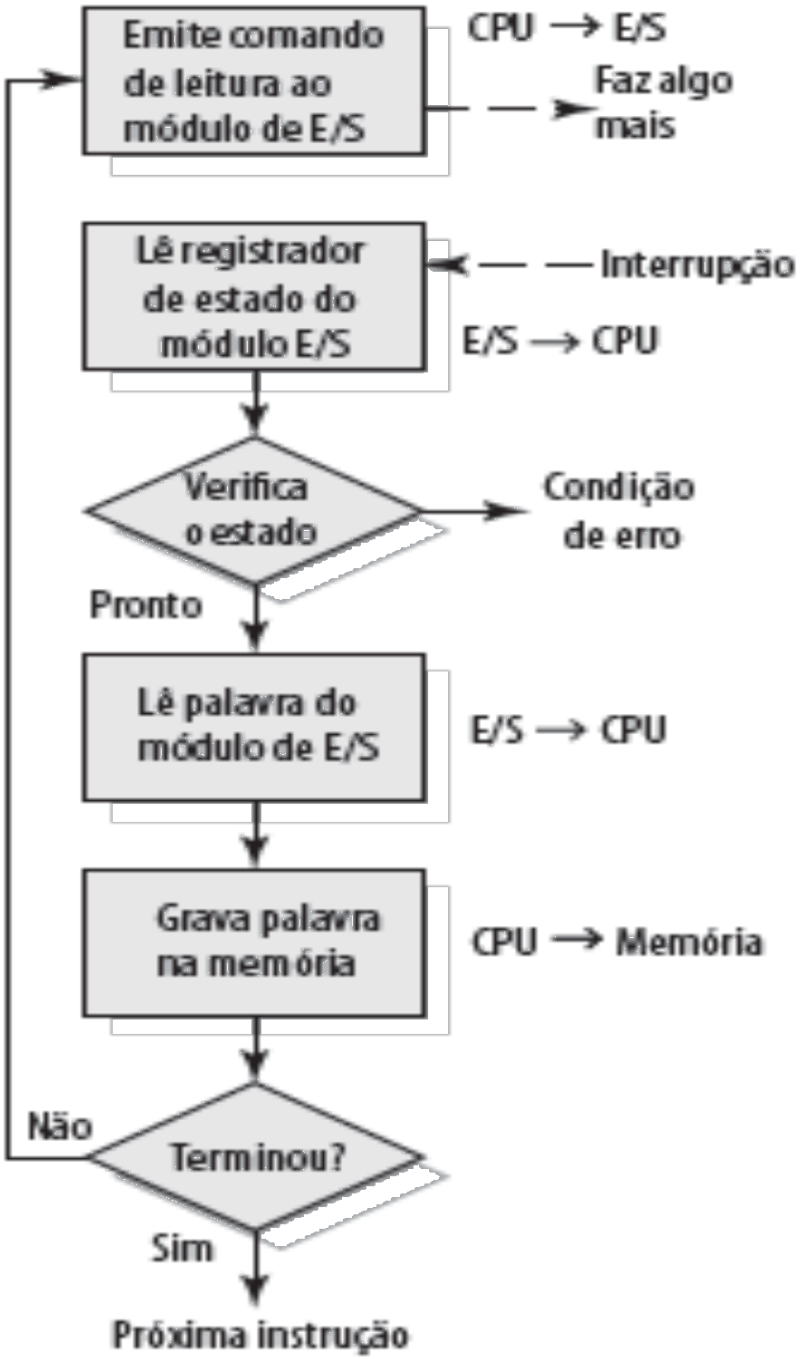
E/S Comandada por Interrupção



E/S Executada por Interrupção

- Evita espera da CPU
- A CPU não verifica o dispositivo diversas vezes
- O módulo de E/S gera uma interrupção quando está pronto

E/S Comandada por Interrupção

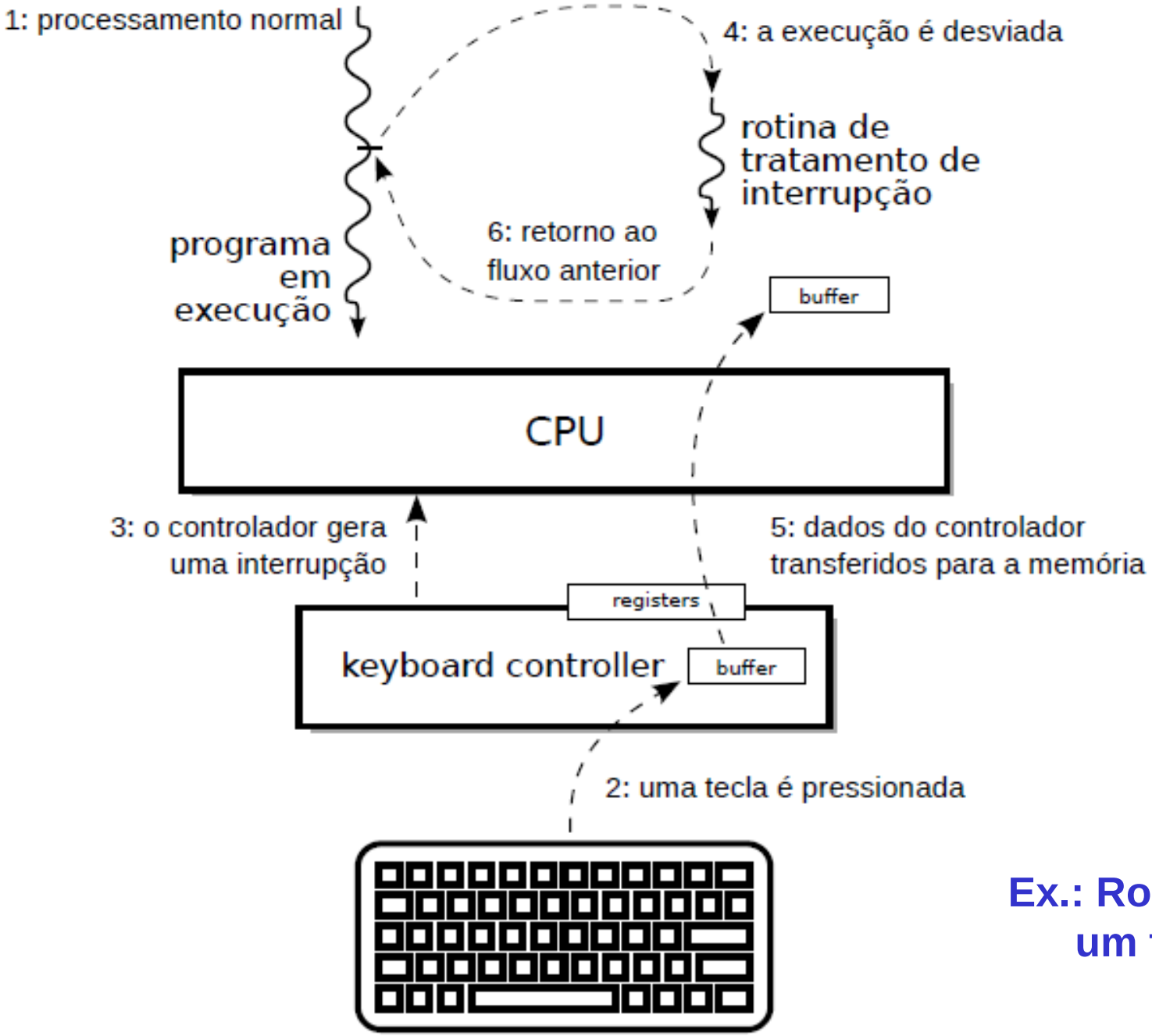


E/S Comandada por Interrupção: Operação Básica

1. A CPU envia comando de leitura
2. O módulo de E/S adquire os dados do periférico enquanto a CPU faz outra coisa
3. O módulo de E/S interrompe a CPU
4. A CPU requisita dados
   - Se não houve erro
5. O módulo de E/S transfere dados

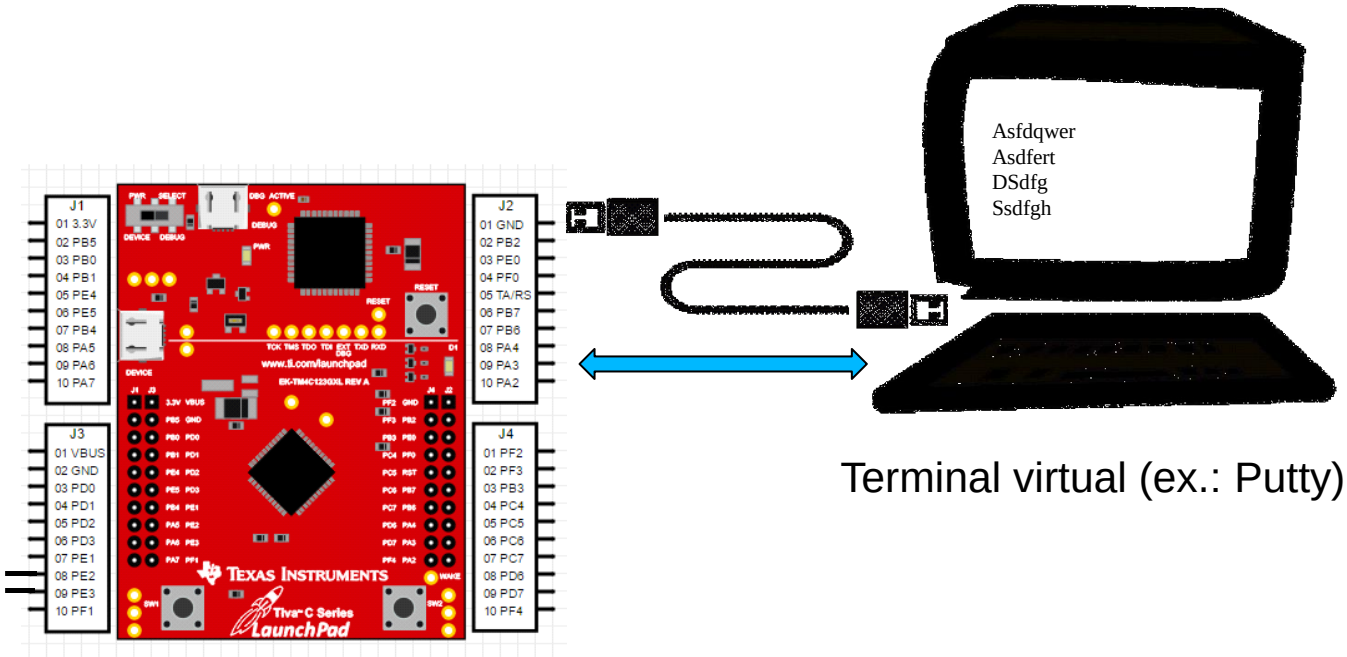
Ponto de Vista da CPU

1. Envia comando de leitura
2. Faz outra coisa...
3. Verifica se ocorreu alguma interrupção ao final de cada ciclo de instrução
4. Caso a interrupção tenha ocorrido:
   - Salva o contexto (registradores)
   - Processa interrupção
     - Por ex.: busca e armazena dados



Ex.: Roteiro típico de um tratamento de interrupção

Exemplo: ISR da UART - Tiva C



Exemplo: ISR da UART - Tiva C

```c
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "driverlib/pin_map.h"
#include "driverlib/sysctl.h"
#include "driverlib/uart.h"

void UARTIntHandler(void)
{
    uint32_t ui32Status;
    ui32Status = UARTIntStatus(UART0_BASE, true); //get interrupt status
    UARTIntClear(UART0_BASE, ui32Status); //clear the asserted interrupts

    while(UARTCharsAvail(UART0_BASE)) //loop while there are chars
    {
        UARTCharPutNonBlocking(UART0_BASE, UARTCharGetNonBlocking(UART0_BASE)); //echo character
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, GPIO_PIN_2); //blink LED
        SysCtlDelay(SysCtlClockGet() / (1000 * 3)); //delay ~1 msec
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0); //turn off LED
    }
}
```

```c
int main(void) {

    SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
```

-
-
-
-

```c
    UARTCharPut(UART0_BASE, 'r');
    UARTCharPut(UART0_BASE, ' ');
    UARTCharPut(UART0_BASE, 'T');
    UARTCharPut(UART0_BASE, 'e');
    UARTCharPut(UART0_BASE, 'x');
    UARTCharPut(UART0_BASE, 't');
    UARTCharPut(UART0_BASE, ':');
    UARTCharPut(UART0_BASE, ' ');

    while (1) //let interrupt handler do the UART echo function
    {
//      if (UARTCharsAvail(UART0_BASE)) UARTCharPut(UART0_BASE, UARTCharGet(UART0_BASE));
    }

}
```

Código principal

Questões de Projeto
E/S Comandada por Interrupção

Questões de Projeto
E/S Comandada por Interrupção

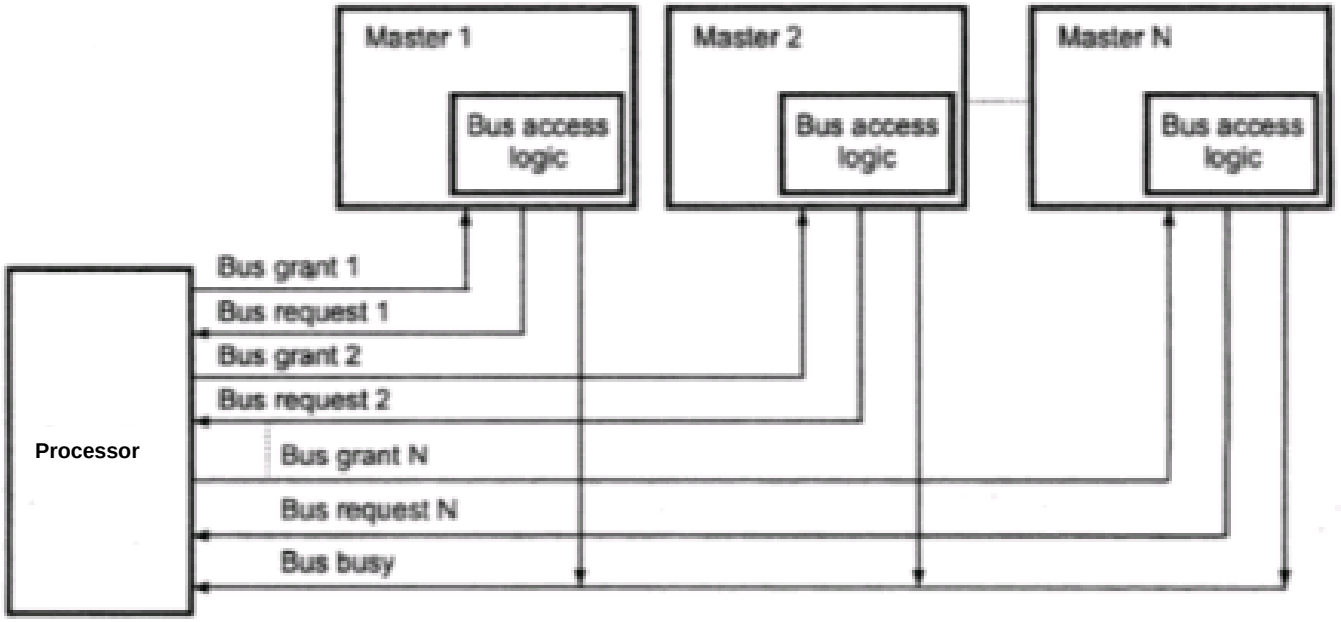
1. Como identificar o módulo que gerou a interrupção?

Questões de Projeto
E/S Comandada por Interrupção

1. Como identificar o módulo que gerou a interrupção?
2. Como lidar com múltiplas interrupções?
   - Definição de prioridades
   - Rotina de tratamento de interrupção sendo interrompida
     - Aninhamento (*Nesting*)

Identificação do Módulo

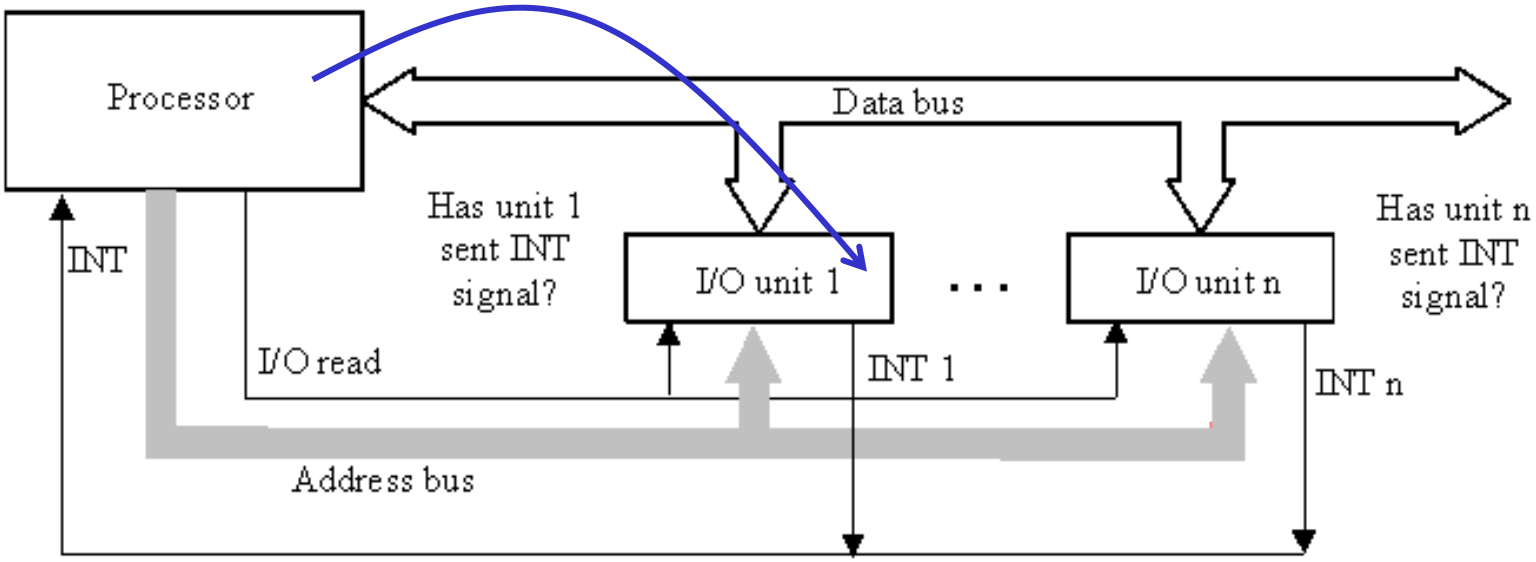
1. Uma linha de interrupção para cada módulo



- Limita o número de dispositivos
- Pois, o número de pinos é limitado

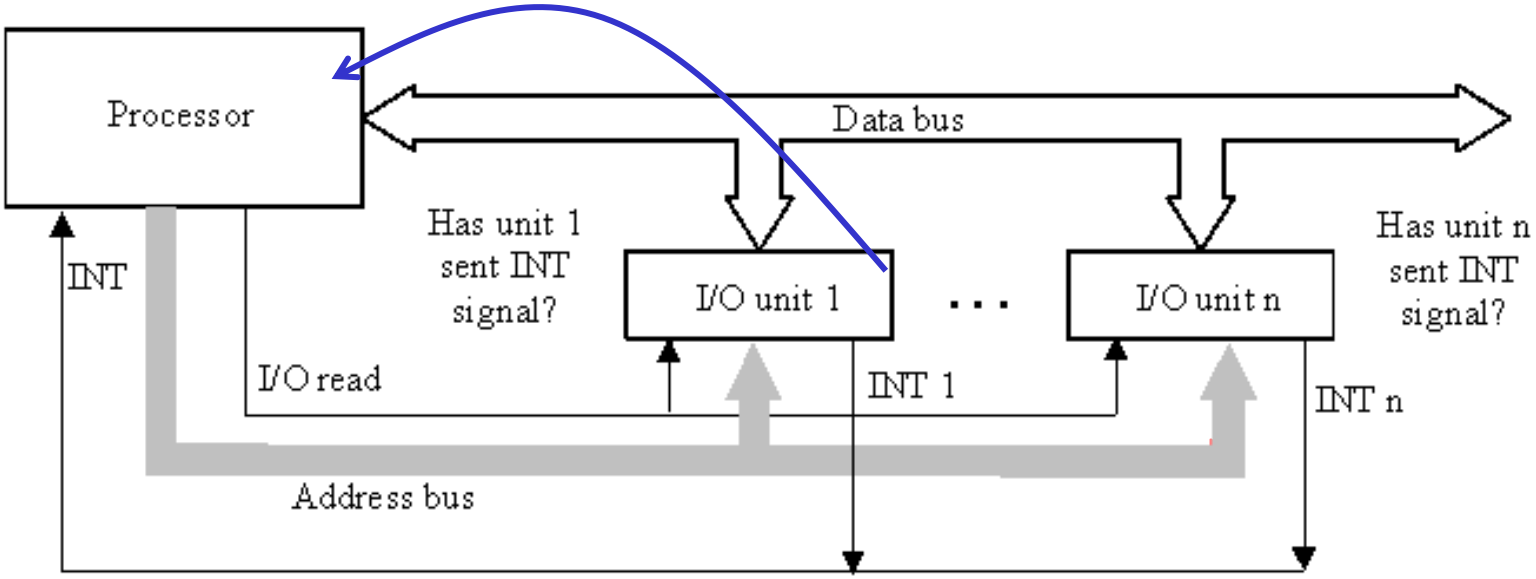
Identificação do Módulo

2. Identificação por SW
   - A CPU pergunta a cada módulo qual foi o gerador da interrupção



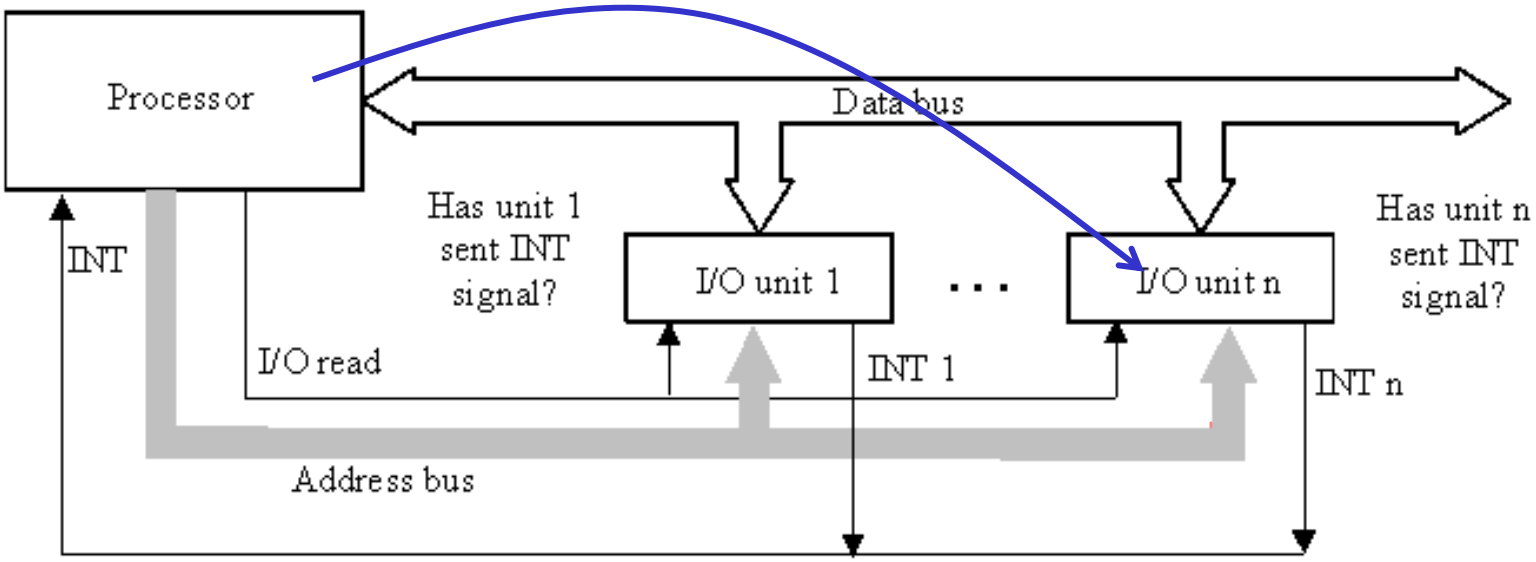
Identificação do Módulo

2. Identificação por SW
   - A CPU pergunta a cada módulo qual foi o gerador da interrupção



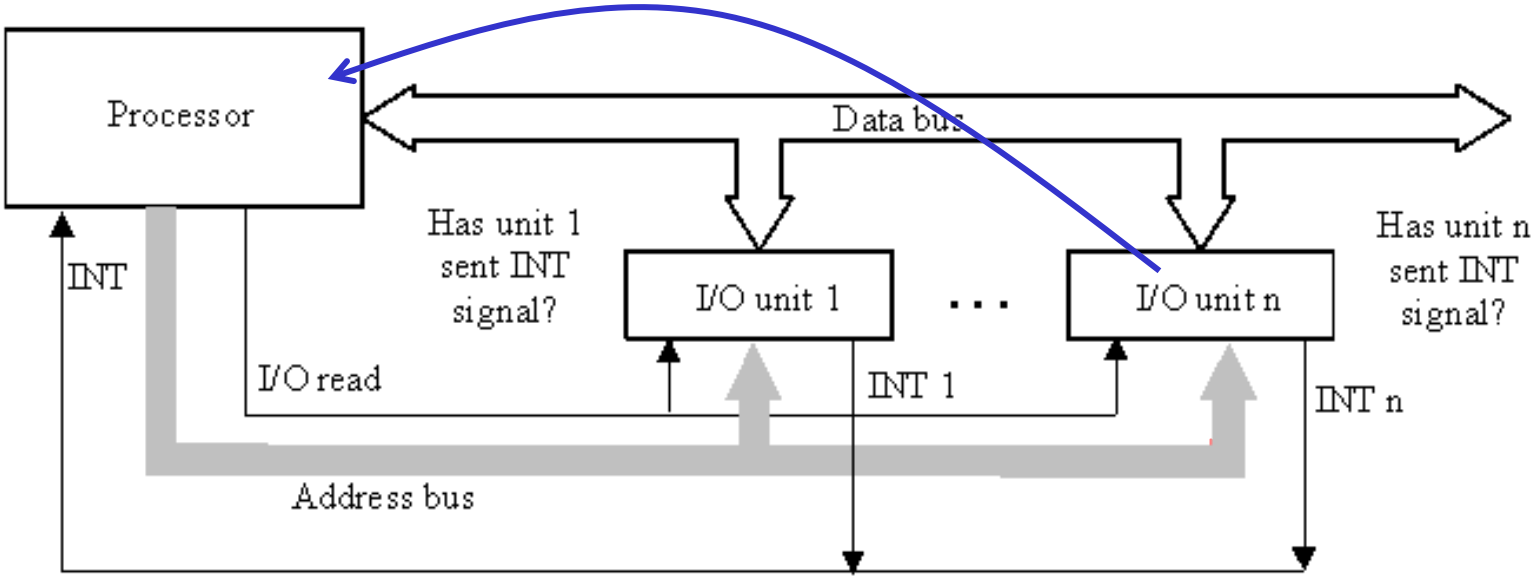
Identificação do Módulo

2. Identificação por SW
   - A CPU pergunta a cada módulo qual foi o gerador da interrupção



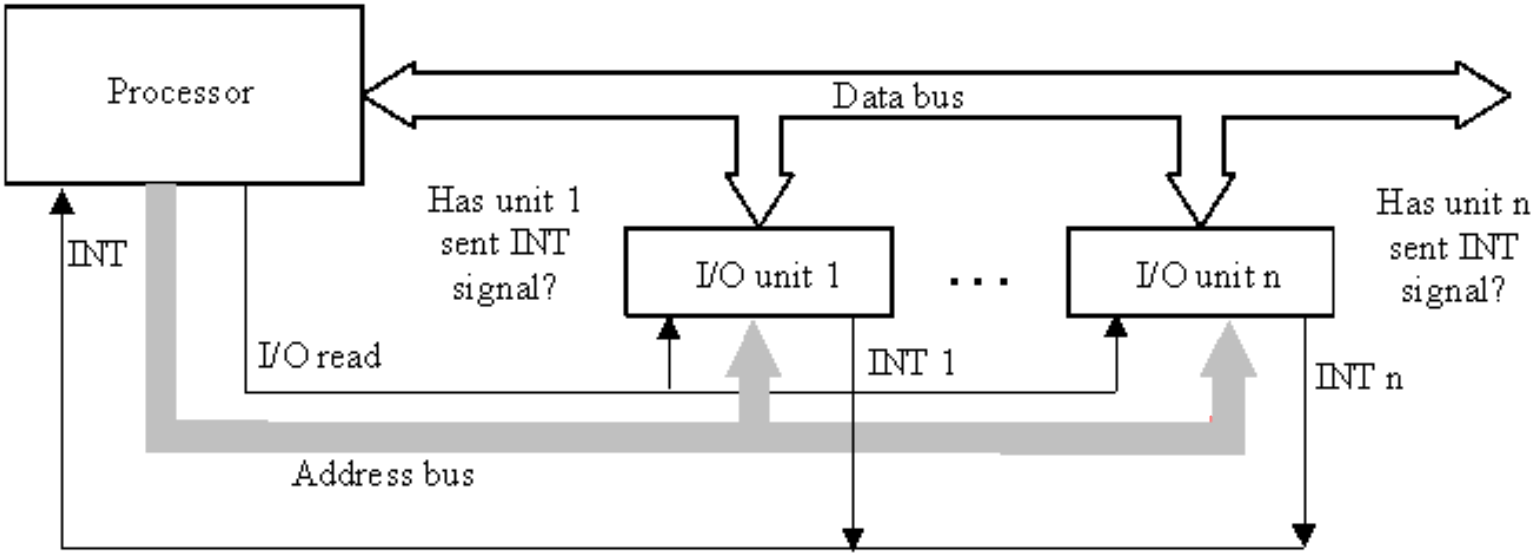
Identificação do Módulo

2. Identificação por SW
   - A CPU pergunta a cada módulo qual foi o gerador da interrupção



Identificação do Módulo

2. Identificação por SW
   - A CPU pergunta a cada módulo qual foi o gerador da interrupção
   - Muito lento!



Identificação do Módulo

3. *Daisy Chain* (cadeia circular) ou Identificação por HW

    1. Sinal de reconhecimento de interrupção enviado através de uma **cadeia circular**

Identificação do Módulo

3. *Daisy Chain* (cadeia circular) ou Identificação por HW

    1. Sinal de reconhecimento de interrupção enviado através de uma **cadeia circular**
    2. O módulo responsável envia o vetor de interrupção (endereço ou identificador da E/S) ao barramento de dados

Identificação do Módulo

3. *Daisy Chain* (cadeia circular) ou Identificação por HW

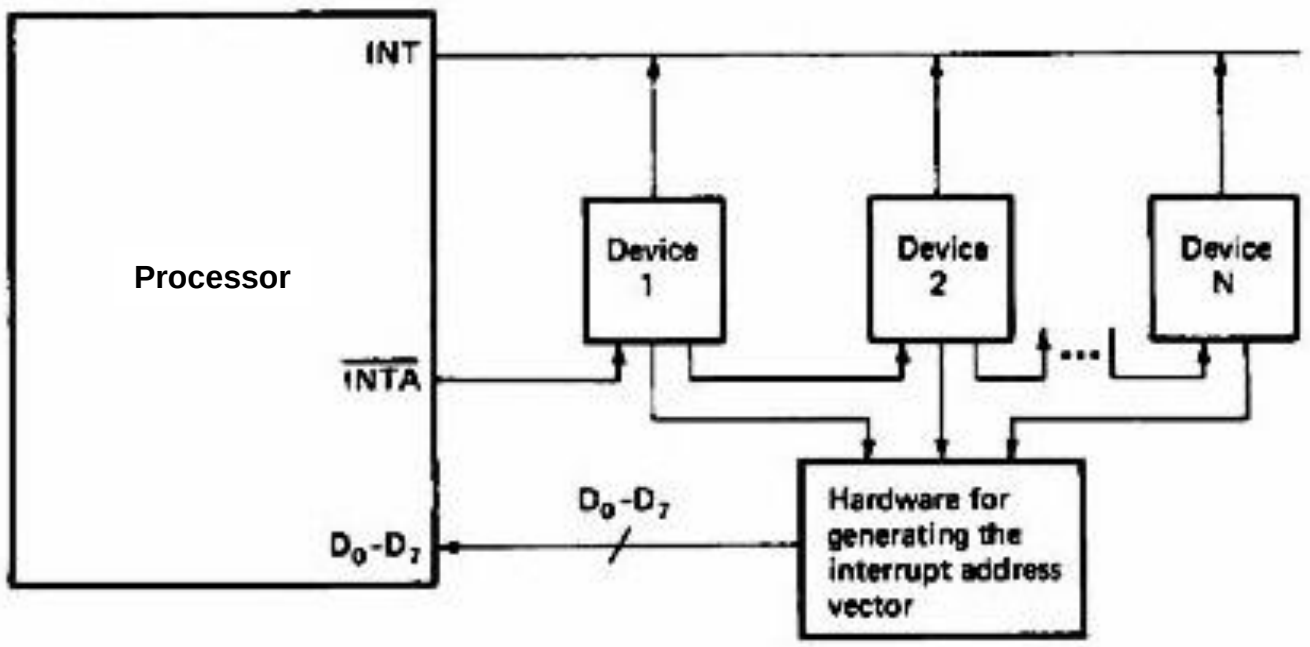
    1. Sinal de reconhecimento de interrupção enviado através de uma **cadeia circular**
    2. O módulo responsável envia o vetor de interrupção (endereço ou identificador da E/S) ao barramento de dados
    3. A CPU usa o vetor para identificar a rotina de tratamento de interrupção

Identificação do Módulo

3. *Daisy Chain* (cadeia circular) ou Identificação por HW

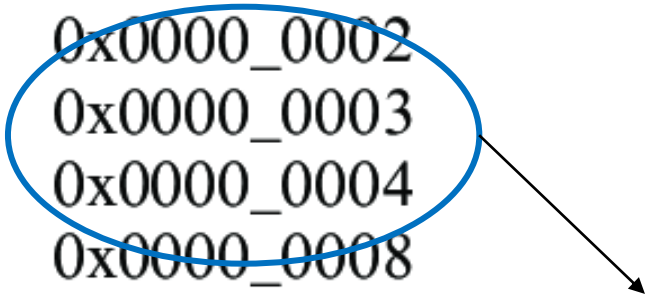
Identificação do Módulo

3. *Daisy Chain* (cadeia circular) ou Identificação por HW



Vetores de interrupção

Fonte da interrupção	Vetor de interrupção
Main program	0x0000_0000
Timer 0	0x0000_0001
Timer 1	0x0000_0002
Multiplier	0x0000_0003
Clock manager	0x0000_0004
UART 0 TX	0x0000_0008
UART 0 RX	0x0000_0009
UART 1 TX	0x0000_000A
UART 1 RX	0x0000_000B
I2C	0x0000_000B
SPI	0x0000_000C
Parallel port	0x0000_000D
External interrupt 1	0x0000_0010
External interrupt 0	0x0000_0018



Endereços de ISR

Identificação do Módulo

4. Bus Master (arbitragem do barramento)
   - Apenas um módulo pode acessar o barramento por vez
     1. O módulo de E/S deve requisitar o barramento antes de gerar uma interrupção
     2. Quando a CPU detecta a interrupção, responde usando a linha de reconhecimento de interrupção
     3. O módulo que causou a interrupção então coloca o vetor de interrupção no barramento de dados
   - ex.: PCI & SCSI

Definição de prioridades

1. Múltiplas linhas

   - Cada linha de interrupção tem uma prioridade
   - Linhas de maior prioridade podem interromper linhas de menor prioridade

Definição de prioridades

2. Identificação por SW

   - A ordem de interrogação dos módulos define as suas prioridades

Definição de prioridades

3. *Daisy Chain* - Identificação por HW

   - A ordem de conexão dos módulos na cadeia define as suas prioridades

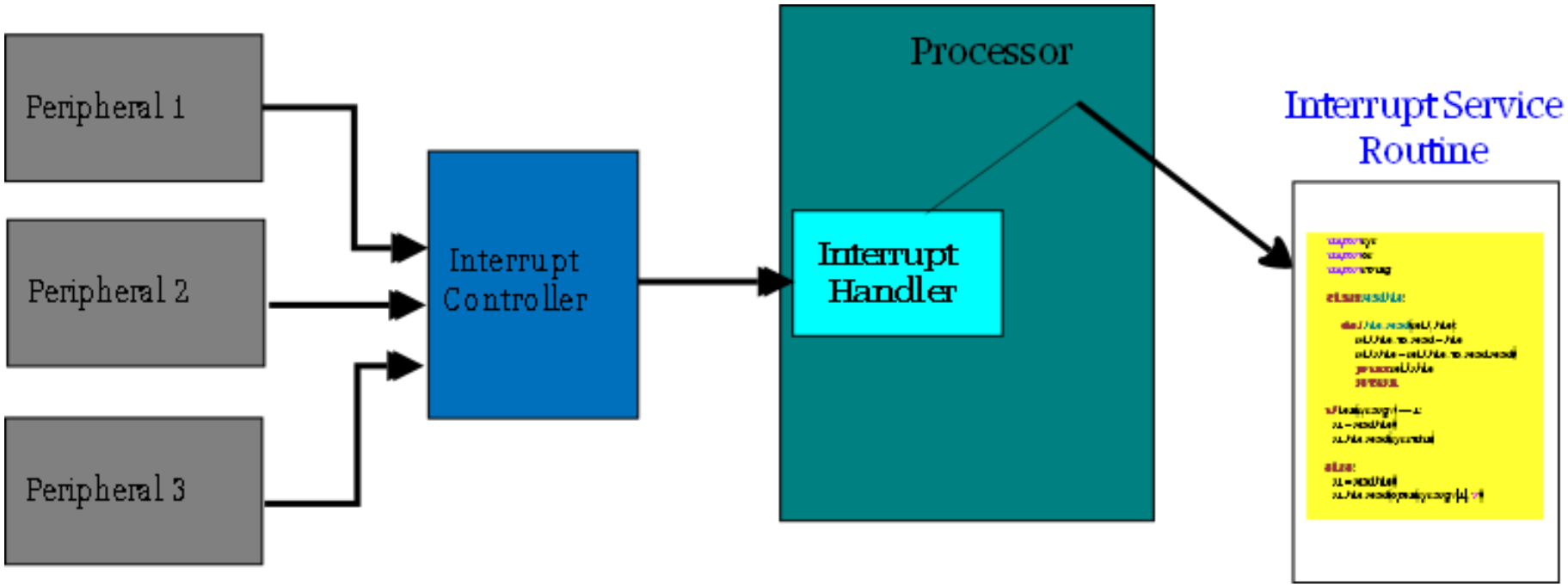
Definição de prioridades

4. Arbitração do barramento

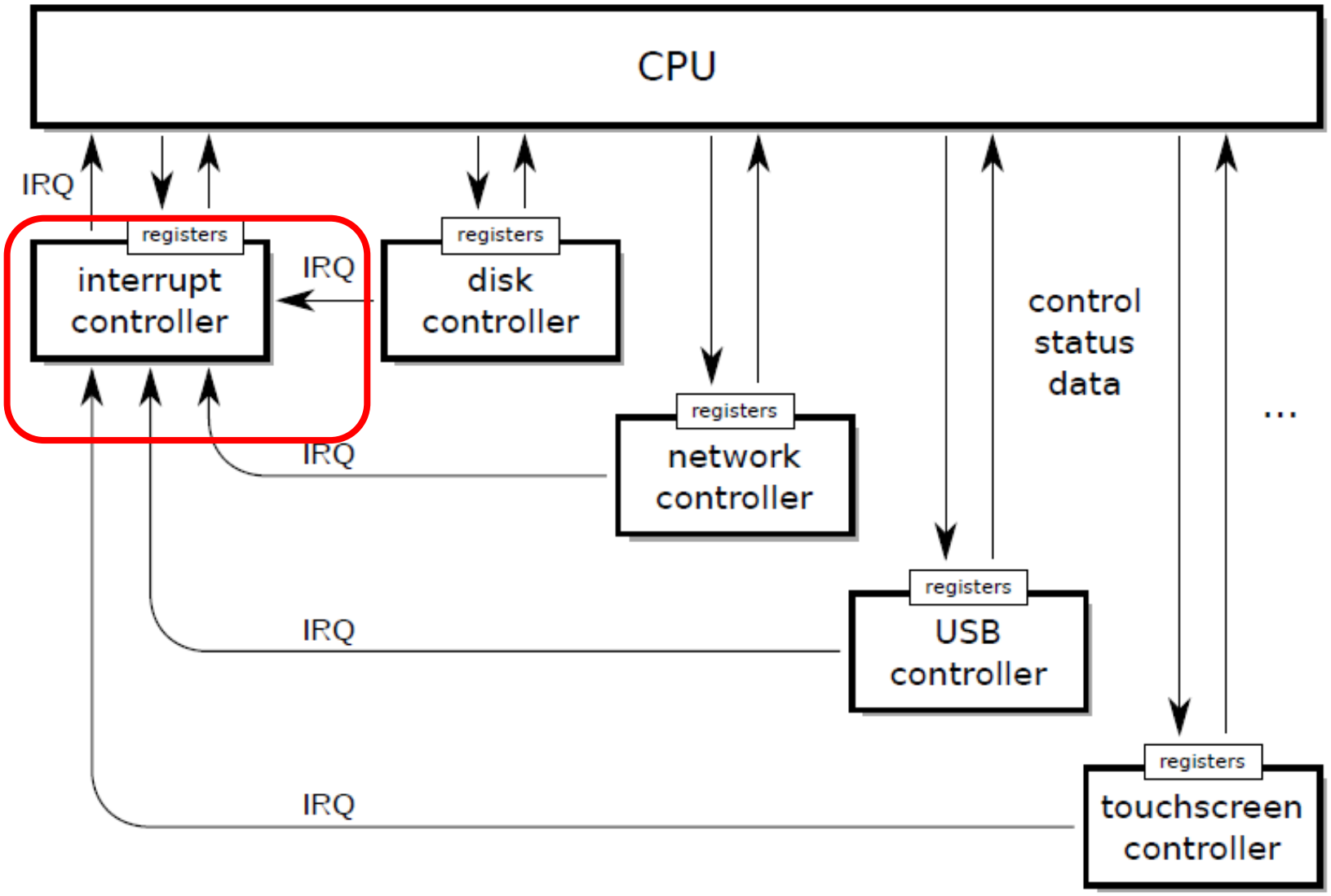
   - Um esquema de prioridades (similar ao definido no acesso ao barramento) pode ser definido

Controladores de Interrupção

Nas arquiteturas *mais complexas*, as interrupções de E/S não são transmitidas diretamente ao processador, e sim a um **controlador de interrupções programável**



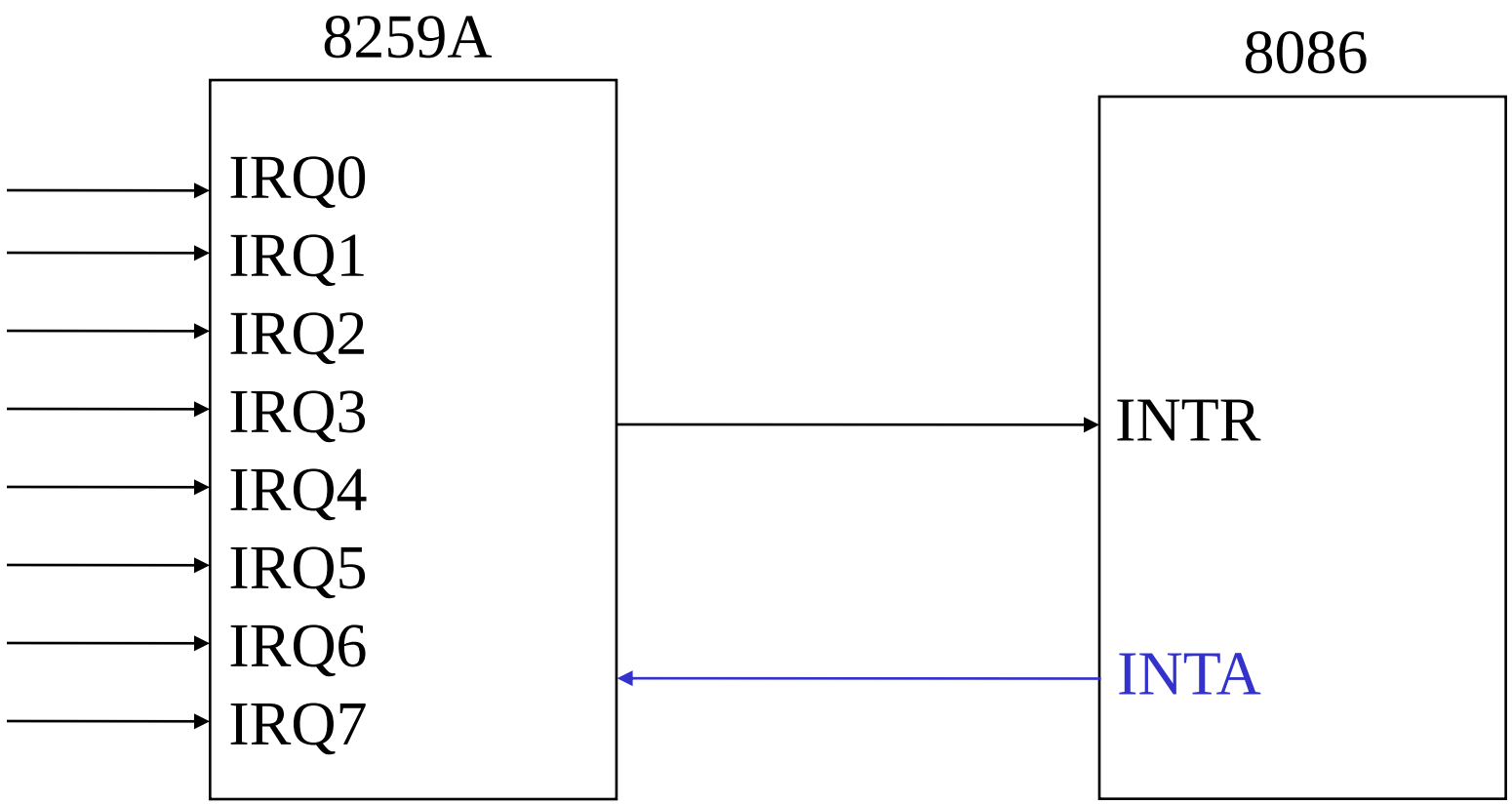
Uso de um controlador de interrupções



Exemplo: Barramento do PC

- O processador **8086** tem uma única linha de interrupção
- Sistemas (simples) baseados no **8086** usam um controlador de interrupção 8259A
- O controlador 8259A tem 8 linhas de interrupção

Esquema de Interrupção do PC

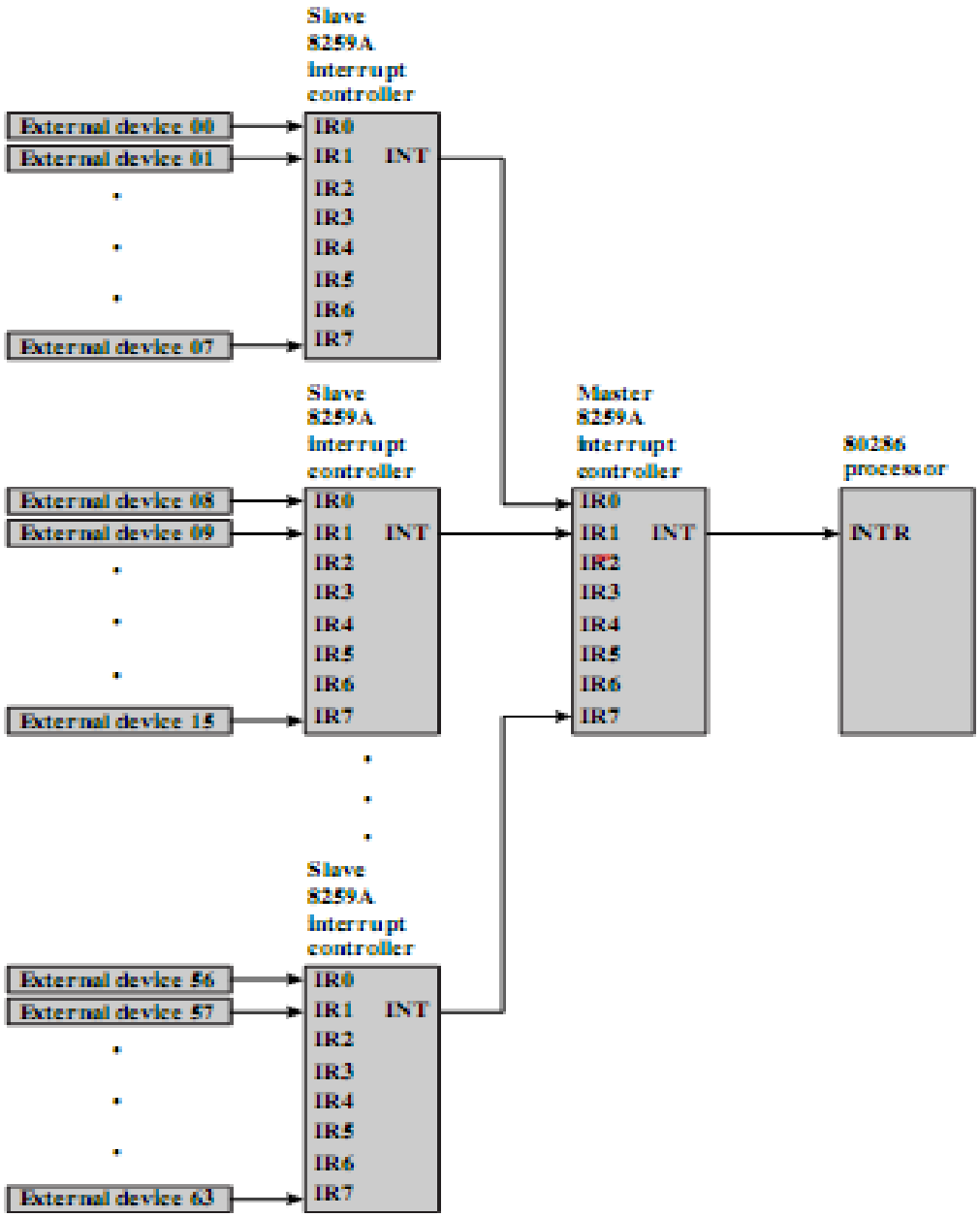


Sequência de Eventos

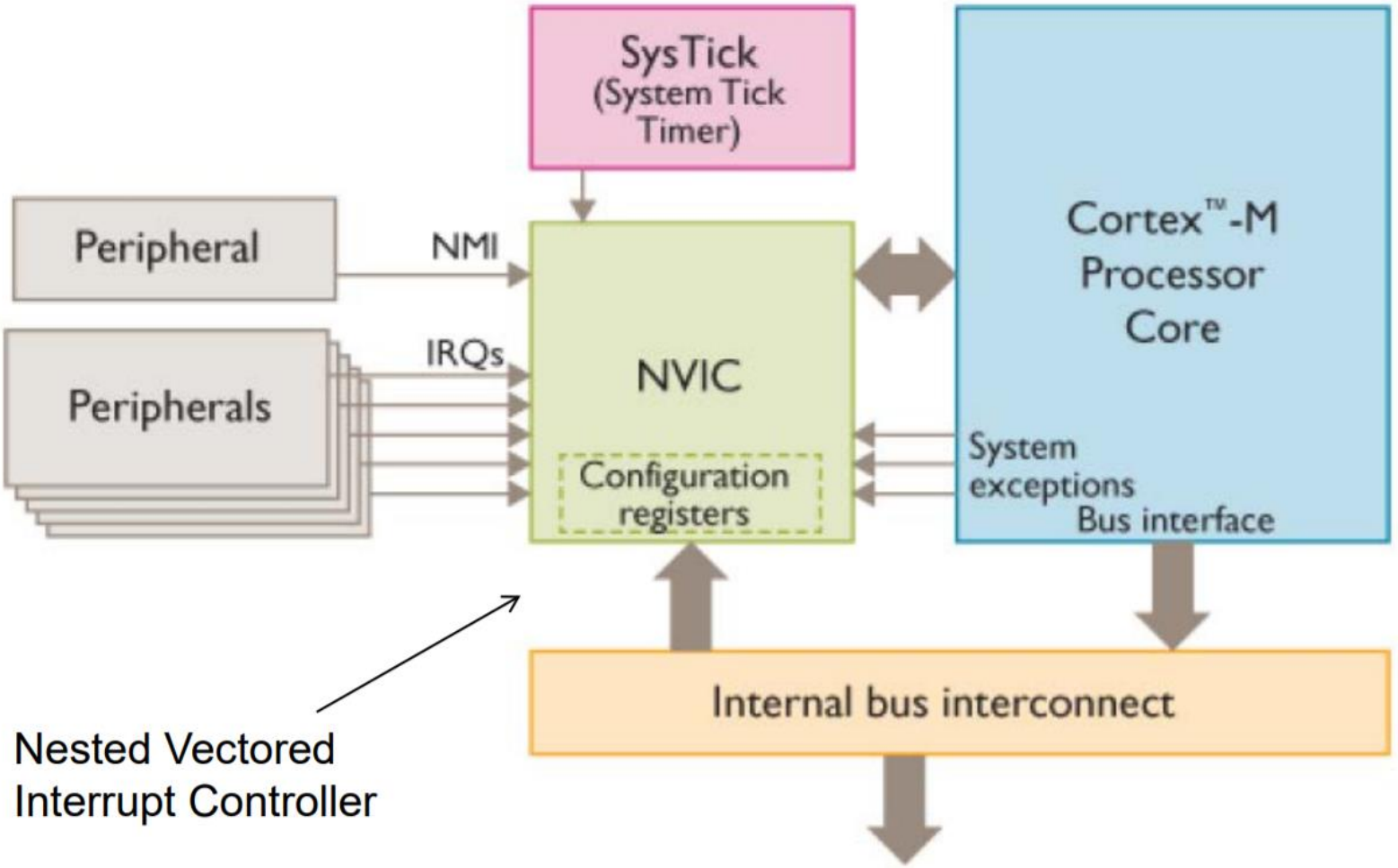
1. 8259A aceita interrupções
2. 8259A determina prioridades
3. 8259A sinaliza para o 8086 (ativação da linha INTR)
4. A CPU acusa que a interrupção ocorreu (INTA)
5. 8259A coloca o vetor de identificação correto no barramento de dados
6. A CPU processa a interrupção

Uso Geral do Controlador 8259A

Quando for necessário controlar mais de 8 módulos pode ser feito um arranjo em cascata para até 64 módulos.



NVIC da Tiva C - ARM

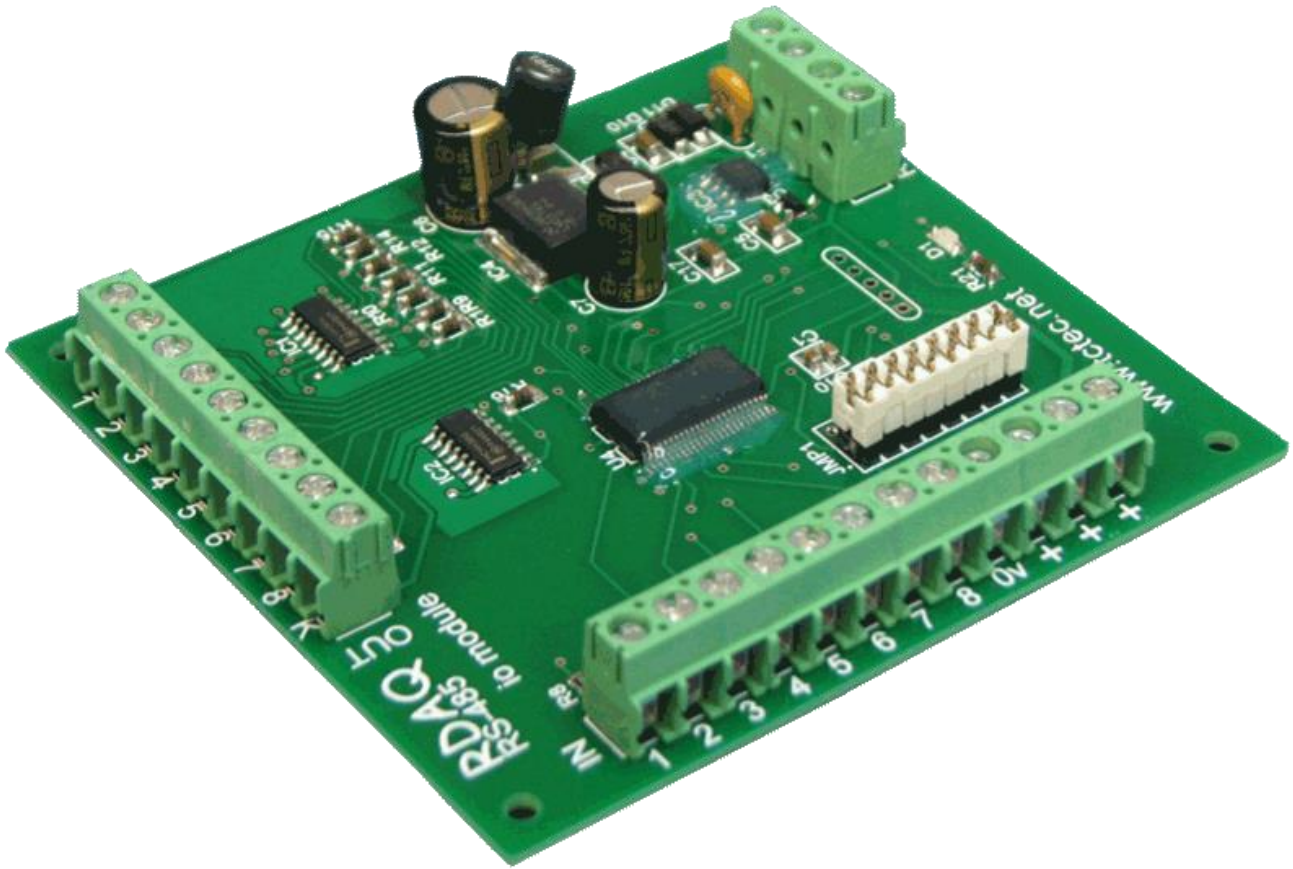


NVIC - Tiva C

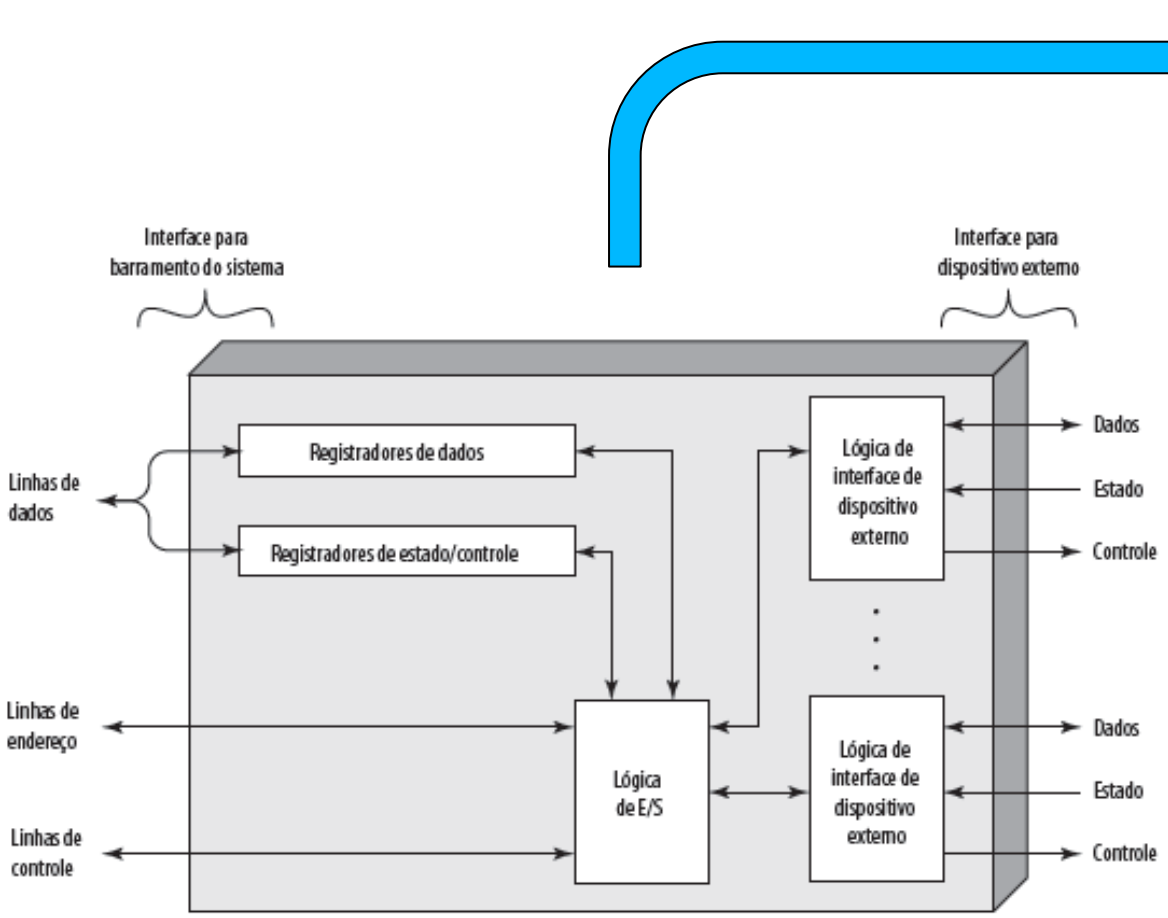
```
EXPORT  __Vectors

__Vectors                               ; address    ISR
        DCD     0                       ; 0x00000020 Reserved
        DCD     0                       ; 0x00000024 Reserved
        DCD     0                       ; 0x00000028 Reserved
        DCD     SVC_Handler             ; 0x0000002C SVCall Handler
        DCD     DebugMon_Handler        ; 0x00000030 Debug Monitor
Handler
        DCD     0                       ; 0x00000034 Reserved
        DCD     PendSV_Handler          ; 0x00000038 PendSV Handler
        DCD     SysTick_Handler         ; 0x0000003C SysTick Handler
        DCD     GPIOPortA_Handler       ; 0x00000040 GPIO Port A
        DCD     GPIOPortB_Handler       ; 0x00000044 GPIO Port B
        DCD     GPIOPortC_Handler       ; 0x00000048 GPIO Port C
        DCD     GPIOPortD_Handler       ; 0x0000004C GPIO Port D
        DCD     GPIOPortE_Handler       ; 0x00000050 GPIO Port E
        DCD     UART0_Handler           ; 0x00000054 UART0
        DCD     UART1_Handler           ; 0x00000058 UART1
        DCD     SSI0_Handler            ; 0x0000005C SSI
        DCD     I2C0_Handler            ; 0x00000060 I2C
        DCD     PWM0Fault_Handler       ; 0x00000064 PWM Fault
        DCD     PWM0Generator0_Handler  ; 0x00000068 PWM 0 Generator
0
        DCD     PWM0Generator1_Handler  ; 0x0000006C PWM 0 Generator
1
        DCD     PWM0Generator2_Handler  ; 0x00000070 PWM 0 Generator
2
        DCD     Quadrature0_Handler     ; 0x00000074 Quadrature Encoder
0
```

Módulo de E/S

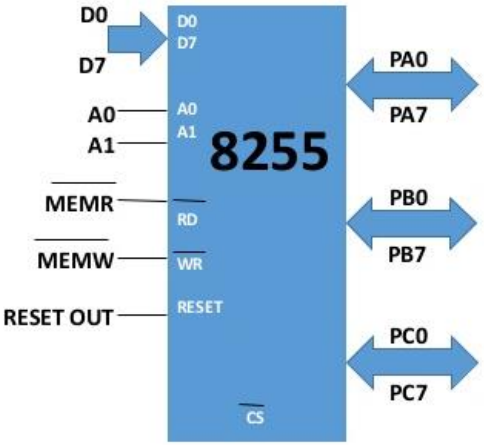


Exemplo de módulo real de E/S



Exemplo de módulo de E/S

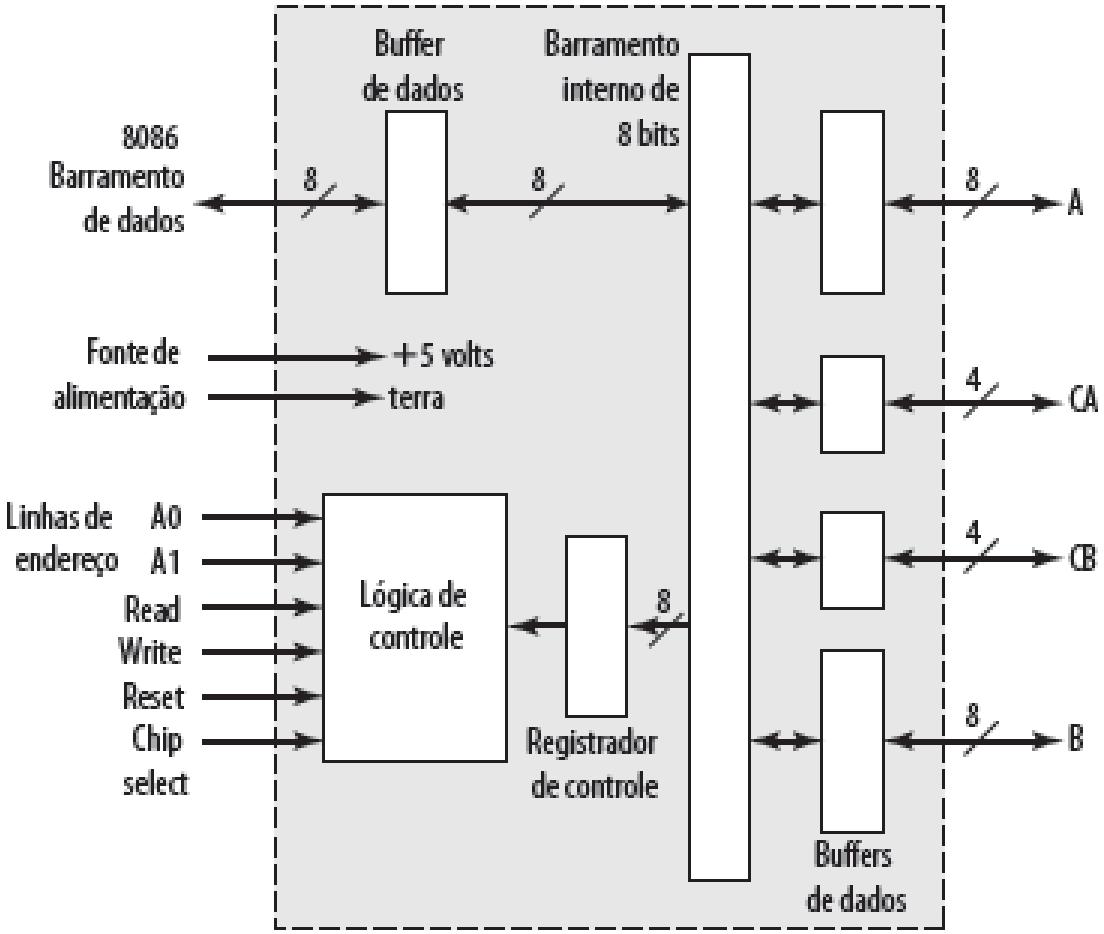
- Intel 8255A – *Programable Peripheral Interface*
- Projetada para o 80386

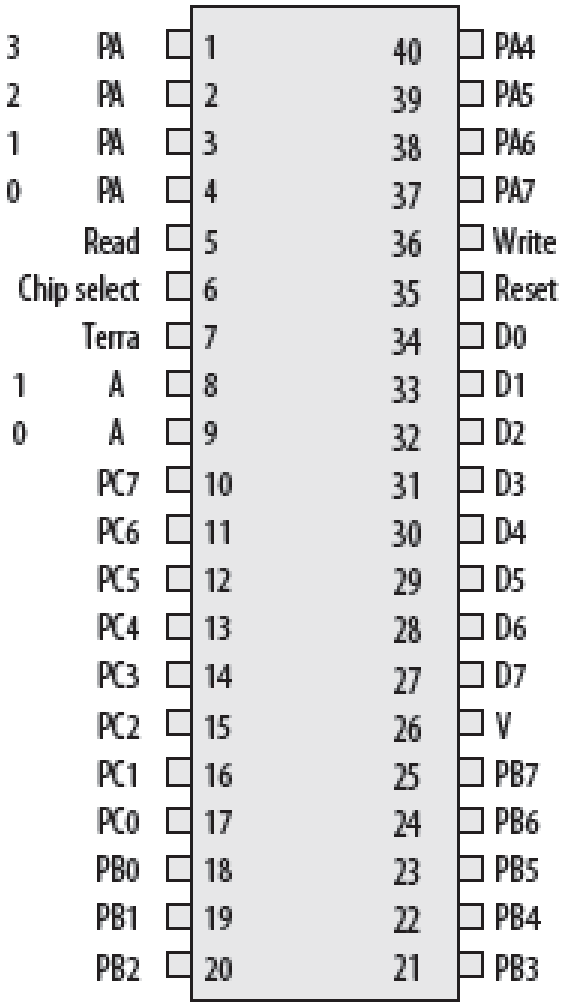


Exemplo concreto

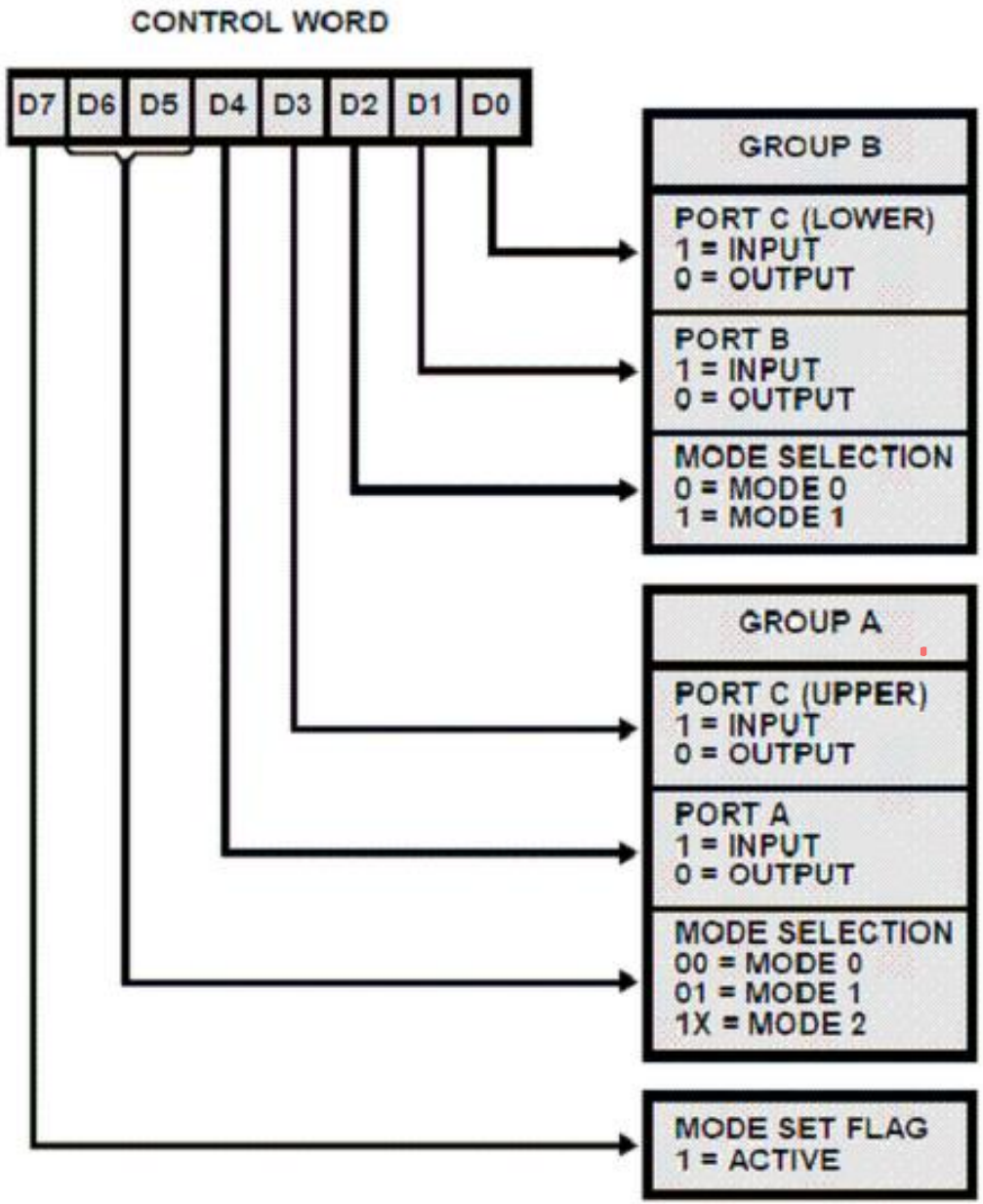
Interface programável de Periféricos

Intel 82C55A

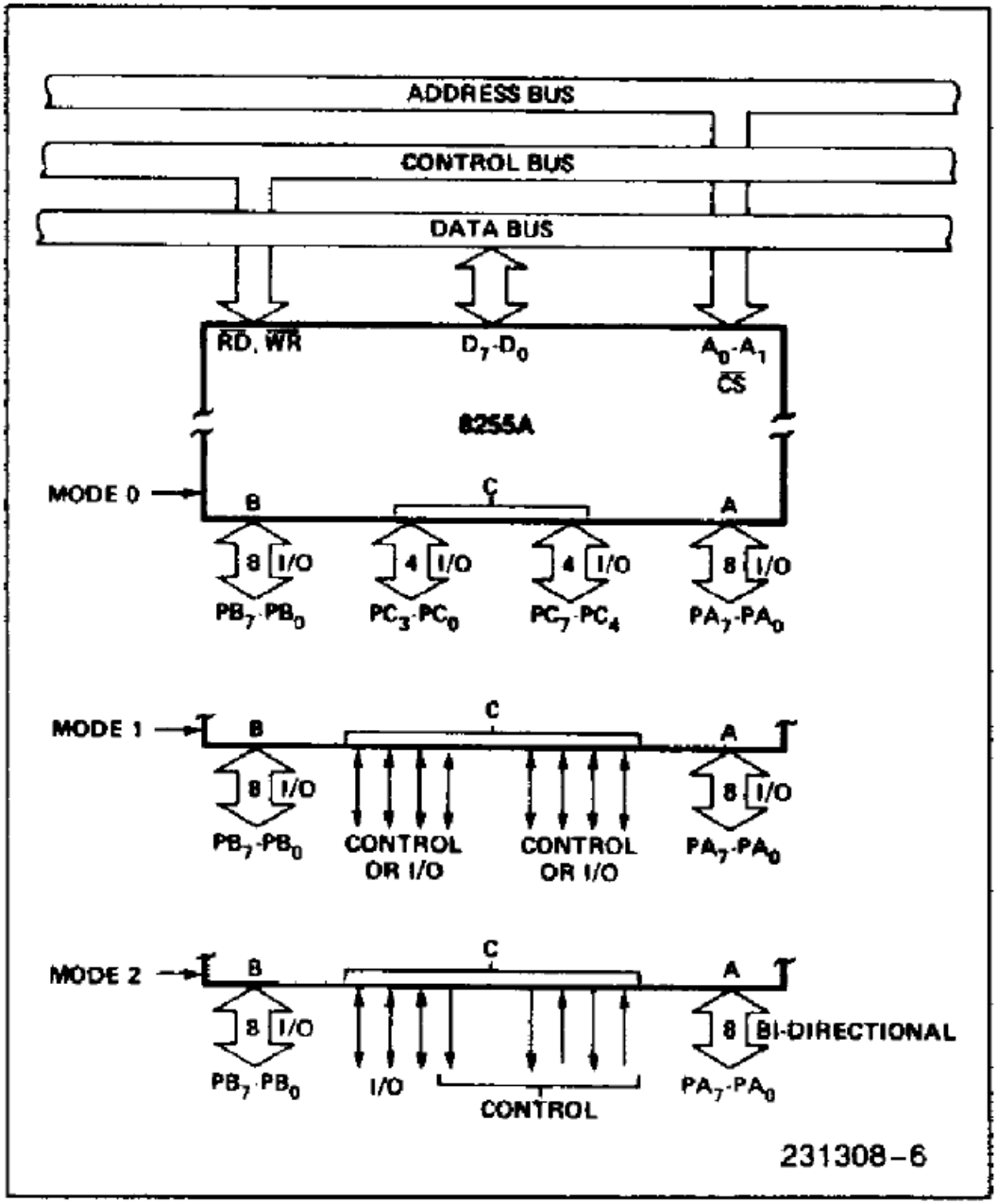




Registrador de Controle

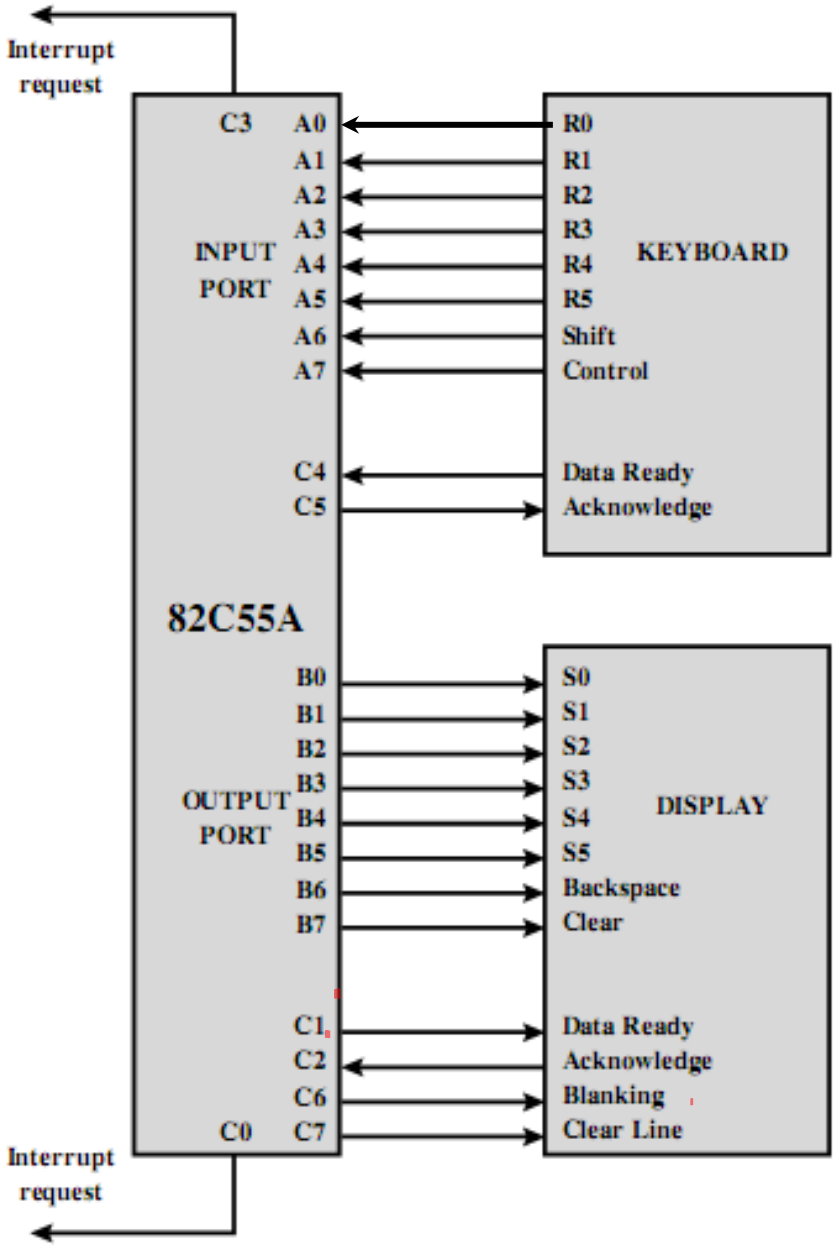


Modos de Operação



Interface de Periféricos 82C55A

Ex.: Interface de teclado/vídeo:



Técnicas de E/S

- Programada
- Executada por Interrupção
- Direct Memory Access (DMA)

Questões

1. Qual a importância dos *buffers* na operação de E/S?
2. O que é um transdutor?
3. Existe(m) caso(s) em que é mais vantajoso usar o *Polling*?
4. Cite exemplos de sinais de controle e de estado trocados durante a operação de E/S.