


Arquitetura de Sistemas Embarcados

Edna Barros (ensb@cin.ufpe.br)

Centro de Informática – UFPE



Introdução

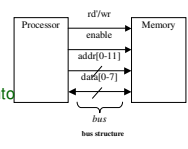
- Aspectos funcionais de sistemas embarcados
 - Processamento
 - Processadores
 - Transformação nos dados
 - Armazenamento
 - Memória
 - Retenção dos dados
 - Comunicação
 - Transferência de informação entre processador e memória
 - Implementação usando Barramentos
 - Denominado: Interfaceamento

Engenharia de Sistemas Embarcados 4

Capítulo 6: Comunicação

Um barramento simples

- Fios:
 - Uni-direcional ou bi-direcional
 - Uma linha pode representar fios múltiplos
- Barramentos
 - Conjunto de fios com função única
 - Barramento de endereço, barramento de dados
 - Ou uma coleção completa de fios
 - Endereço, dados e controle
 - Protocolo associado: regras para a comunicação



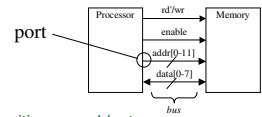
Engenharia de Sistemas Embarcados 5

Roteiro

- Conceitos básicos
- Interface do Microprocessador
 - Endereçamento de E/S
 - Interrupções
 - Acesso direto à memória
- Arbitragem
- Barramentos Hierarquicos
- Protocolos
 - Serial
 - Paralelo
 - Wireless

Engenharia de Sistemas Embarcados 3

Portas



- Interliga dispositivo ao ambiente
- Conecta barramento a processador ou memória
- Geralmente conhecido como pino
 - Pinos na periferia de um IC que são plugados na placa de circuito impresso
 - Presente: "pads" de metal conectando processadores e memória em um único IC
- Fio único ou conjunto de fios com função única
 - EX. Porta de endereços com 12 fios

Engenharia de Sistemas Embarcados 6

Diagramas Temporais

- Método comum de descrever protocolo de comunicação
- Sinais de controle: baixo ou alto
 - Pode ser ativados em "0" ou "1"
- Sinais de dados: válidos ou não válidos
- Protocolos podem ter subprotocolos
 - Ciclo de barramento: read e write
 - Cada com a duração de vários ciclos de clock
- Exemplo de Leitura
 - rd/wr ativado em "0" low, endereço disponibilizado em $addr$ pelo tempo mínimo de t_{setup} antes que $enable$ seja ativado, $enable$ causa a memória disponibilizar os dados em $data$ wires durante o tempo t_{read}

Engenharia de Sistemas Embarcados 7

Protocolo de compromisso entre strobe/handshake

- Master asserts req to receive data
- Servant puts data on bus **within time** t_{access}
- Servant can't put data within t_{access} , asserts $wait$ ack (wait line is unused)
- Master receives data and deasserts req
- Servant ready for next request

Fast-response case Slow-response case

Engenharia de Sistemas Embarcados 10

Conceitos básicos de Protocolos

- Atores: mestre inicia, escravo responde
- Direção: determina quem vai ser transmissor e receptor
- Endereços: tipo especial de dado
 - Especifica a localização na memória, um periférico ou um registrador em um periférico
- Multiplexação no Tempo
 - Compartilhamento de um conjunto de fios para múltiplos tipos de dados
 - Economia de fios, retardo maior

Engenharia de Sistemas Embarcados 8

ISA bus protocol – memory access

ISA: Industry Standard Architecture

- Comum nos processadores 80x86's
- Características
 - Endereço: 20-bit
 - Protocolo de compromisso entre strobe/handshake
 - Default: 4 cycles
 - Se $CHRDY$ não for desativada – ciclos adicionais de wait (até 6)

Engenharia de Sistemas Embarcados 11

Conceitos básicos de protocolos: métodos de controle

Strobe protocol

- Master asserts req to receive data
- Servant puts data on bus **within time** t_{access}
- Master receives data and deasserts req
- Servant ready for next request

Handshake protocol

- Master asserts req to receive data
- Servant puts data on bus **and asserts** ack
- Master receives data and deasserts req
- Servant ready for next request

Engenharia de Sistemas Embarcados 9

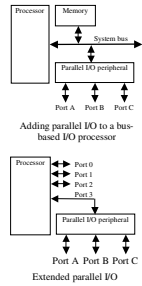
Interfaceando o Microprocessador: Endereçamento de E/S

- Um processador se comunica com outros dispositivos através de alguns de seus pinos
 - E/S baseada em Portas (E/S paralela)
 - Processador tem uma ou mais portas de N-bits
 - Software lê e escreve nas portas como escreve em registradores
 - E.x., $P0 = 0xFF$; $v = P1.2$; -- $P0$ e $P1$ são portas de 8-bits
 - E/S baseada em Barramento
 - Processador possui portas de endereço, dados e controle que formam um barramento
 - Protocolo de comunicação é embutido no processador
 - Uma única instrução para operação de leitura ou escrita

Engenharia de Sistemas Embarcados 12

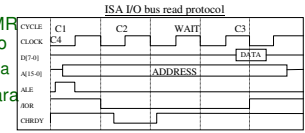
Compromissos/extensões

- **Periférico de E/S Paralela**
 - Quando o processador só suporta E/S baseada em barramento e necessita de E/S paralela
 - Cada porta no periférico é conectada a registrador no periférico que é lido ou escrito pelo processador
- **E/S Paralela Estendida**
 - Quando o processador suporta E/S baseada em portas e precisa de mais portas
 - Uma ou mais portas se liga ao periférico de extensão de portas.
 - e.x., extensão de 4 portas para 6 portas



ISA bus

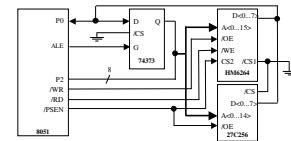
- ISA suporta E/S padrão
- I/O
 - /IOR diferencia de /MEMP para leitura em periférico
 - /IOW usado para escrita
 - Endereços de 16-bits para E/S e de 20 bits para memória
 - Similar ao protocolo de memória



Tipos de E/S baseada em Barramento

- **Processador se comunica com memória e periféricos usando o mesmo barramento**
 - **E/S Mapeada em Memória**
 - Registradores de periféricos ocupam endereços no mesmo espaço de endereçamento da memória
 - e.x., Barramento tem endereço de 16-bits
 - 32K endereços baixos correspondem a memória
 - 32k endereços altos correspondem aos periféricos
 - **E/S Padrão**
 - Pinos adicionais (M/I/O) no barramento indica se o acesso é na memória ou no periférico
 - e.x., Barramento tem endereço de 16-bits
 - Todos os 64K endereços correspondem a memória quando M/I/O é setado para 0
 - Todos os 64K endereços correspondem a periféricos quando M/I/O é setado para 1

Um protocolo de memória simples



- **Interface entre o 8051 e memória externa**
 - Portas P0 e P2 suportam E/S baseada em portas quando memória interna está sendo usada
 - Estas portas servem como barramento para dados/endereços quando memória externa está sendo usada
 - Endereços de 16-bits e dados de 8-bits são multiplexados no tempo; 8-bits do endereço (menos significativos) são armazenados com a ativação do sinal ALE

E/S mapeada em memória vs. E/S Padrão

- **E/S Mapeada em Memória**
 - Não requer instruções especiais
 - Instruções assembly de acesso à memória podem ser usadas para periféricos
 - E/S Padrão requer instruções especiais
- **E/S Padrão**
 - Nenhuma perda de memória para endereçar periféricos
 - Decodificação de endereços em periféricos mais simples
 - Quando número de periféricos é pequeno, bits mais significativos do endereço podem ser desconsiderados

Interface com o microprocessador: Interrupções

- **Suponha que um periférico receba dados de forma intermitente, que devem ser tratados pelo processador**
 - O processador pode fazer polling no periférico regularmente para testar a chegada de dado
 - O periférico pode interromper o processador quando chegou um dado
- **Necessidade de pino(s) extras: Int**
 - Se Int é igual a 1, o processador suspende o programa em execução e desvia para a rotina de tratamento da interrupção (ISR)
 - Processo conhecido como interrupt-driven I/O
 - "polling" do pino de interrupção está embutido no hardware

Interface com o microprocessador: Interrupções

- Qual o endereço da ISR?
 - Interrupção fixa
 - Endereço embutido no processador e não pode ser alterado
 - O endereço ou uma instrução de desvio estarão armazenados
 - Interrupção Vetorizada
 - Periféricos devem fornecer o endereço
 - Comum quando processador tem vários periféricos conectados ao barramento de sistema
 - Compromisso: Tabela de endereços de interrupção

Engenharia de Sistemas Embarcados 19

Interrupt-driven I/O usando endereço fixo da ISR

2: P1 asserts *Int* to request servicing by the microprocessor

Program memory
 ISR
 16: MOV R0, 0x8000
 17: # modifies R0
 18: MOV 0x8001, R0
 19: RETI # ISR return
 ...
 Main program
 ...
 100: instruction
 101: instruction

Data memory
 System bus
 P1
 0x8000
 P2
 0x8001

µP
 Int
 PC
 100

Engenharia de Sistemas Embarcados 22

Interrupt-driven I/O usando endereço fixo da ISR

1(a): µP is executing its main program.

1(b): P1 receives input data in a register with address 0x8000.

2: P1 asserts *Int* to request servicing by the microprocessor.

3: After completing instruction at 100, µP sees *Int* asserted, saves the PC's value of 100, and sets PC to the ISR fixed location of 16.

4(a): The ISR reads data from 0x8000, modifies the data, and writes the resulting data to 0x8001.

4(b): After being read, P1 deasserts *Int*.

5: The ISR returns, thus restoring PC to 100+1=101, where µP resumes executing.

Engenharia de Sistemas Embarcados 20

Interrupt-driven I/O usando endereço fixo da ISR

3: After completing instruction at 100, µP sees *Int* asserted, saves the PC's value of 100, and sets PC to the ISR fixed location of 16.

Program memory
 ISR
 16: MOV R0, 0x8000
 17: # modifies R0
 18: MOV 0x8001, R0
 19: RETI # ISR return
 ...
 Main program
 ...
 100: instruction
 101: instruction

Data memory
 System bus
 P1
 0x8000
 P2
 0x8001

µP
 Int
 PC
 100

Engenharia de Sistemas Embarcados 23

Interrupt-driven I/O usando endereço fixo da ISR

1(a): µP is executing its main program

1(b): P1 receives input data in a register with address 0x8000.

Program memory
 ISR
 16: MOV R0, 0x8000
 17: # modifies R0
 18: MOV 0x8001, R0
 19: RETI # ISR return
 ...
 Main program
 ...
 100: instruction
 101: instruction

Data memory
 System bus
 P1
 0x8000
 P2
 0x8001

µP
 Int
 PC
 100

Engenharia de Sistemas Embarcados 21

Interrupt-driven I/O usando endereço fixo da ISR

4(a): The ISR reads data from 0x8000, modifies the data, and writes the resulting data to 0x8001.

4(b): After being read, P1 deasserts *Int*.

Program memory
 ISR
 16: MOV R0, 0x8000
 17: # modifies R0
 18: MOV 0x8001, R0
 19: RETI # ISR return
 ...
 Main program
 ...
 100: instruction
 101: instruction

Data memory
 System bus
 P1
 0x8000
 P2
 0x8001

µP
 Int
 PC
 100

Engenharia de Sistemas Embarcados 24

Interrupt-driven I/O usando endereço fixo da ISR

5: The ISR returns, thus restoring PC to $100+1=101$, where μP resumes executing.

Program memory
 ISR
 16: MOV R0, 0x8000
 17: # modifies R0
 18: MOV 0x8001, R0
 19: RETI # ISR return
 ...
 Main program
 100: instruction
 101: instruction

μP
 Data memory
 System bus
 P1 0x8000
 P2 0x8001
 PC 101

Engenharia de Sistemas Embarcados 25

Interrupt-driven I/O usando interrupção vetorizada

2: P1 asserts *Int* to request servicing by the microprocessor

Program memory
 ISR
 16: MOV R0, 0x8000
 17: # modifies R0
 18: MOV 0x8001, R0
 19: RETI # ISR return
 ...
 Main program
 100: instruction
 101: instruction

μP
 Data memory
 System bus
 P1 0x8000
 P2 0x8001
 PC 100

Engenharia de Sistemas Embarcados 28

Interrupt-driven I/O usando interrupção vetorizada

1(a): μP is executing its main program.
 1(b): P1 receives input data in a register with address 0x8000.
 2: P1 asserts *Int* to request servicing by the microprocessor.
 3: After completing instruction at 100, μP sees *Int* asserted, saves the PC's value of 100, and asserts *Inta*.
 4: P1 detects *Inta* and puts interrupt address vector 16 on the data bus.
 5(a): μP jumps to the address on the bus (16). The ISR there reads data from 0x8000, modifies the data, and writes the resulting data to 0x8001.
 5(b): After being read, P1 deasserts *Int*.
 6: The ISR returns, thus restoring PC to $100+1=101$, where μP resumes executing.

Engenharia de Sistemas Embarcados 26

Interrupt-driven I/O usando interrupção vetorizada

3: After completing instruction at 100, μP sees *Int* asserted, saves the PC's value of 100, and asserts *Inta*

Program memory
 ISR
 16: MOV R0, 0x8000
 17: # modifies R0
 18: MOV 0x8001, R0
 19: RETI # ISR return
 ...
 Main program
 100: instruction
 101: instruction

μP
 Data memory
 System bus
 P1 16
 P2 0x8001
 PC 100

Engenharia de Sistemas Embarcados 29

Interrupt-driven I/O usando interrupção vetorizada

1(a): P is executing its main program
 1(b): P1 receives input data in a register with address 0x8000.

Program memory
 ISR
 16: MOV R0, 0x8000
 17: # modifies R0
 18: MOV 0x8001, R0
 19: RETI # ISR return
 ...
 Main program
 100: instruction
 101: instruction

μP
 Data memory
 System bus
 P1 16
 P2 0x8001
 PC 100

Engenharia de Sistemas Embarcados 27

Interrupt-driven I/O usando interrupção vetorizada

4: P1 detects *Inta* and puts interrupt address vector 16 on the data bus

Program memory
 ISR
 16: MOV R0, 0x8000
 17: # modifies R0
 18: MOV 0x8001, R0
 19: RETI # ISR return
 ...
 Main program
 100: instruction
 101: instruction

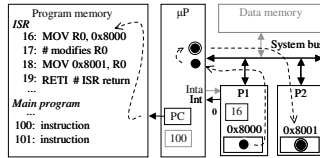
μP
 Data memory
 System bus
 P1 16
 P2 0x8001
 PC 100

Engenharia de Sistemas Embarcados 30

Interrupt-driven I/O usando interrupção vetorizada

5(a): PC jumps to the address on the bus (16). The ISR there reads data from 0x8000, modifies the data, and writes the resulting data to 0x8001.

5(b): After being read, P1 deasserts *Int*.

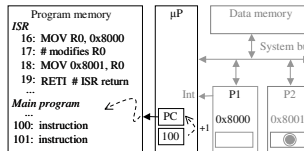


Características adicionais das interrupções

- Interrupções mascaráveis vs. Não mascaráveis
 - Mascarável: programador pode setar bit que faz processador ignorar interrupção
 - Não-mascarável: um pino separado que não pode ser mascarado
- Desvio para ISR
 - Alguns processadores tratam como chamada de subrotina
 - Salvamento completo do estado (PC, registradores) – durar muitos ciclos
 - Outros salvam estado parcial (somente PC)
 - ISR não modificam registradores (se modificar devem salvar antes)
 - Programador deve estar ciente de quais registradores estão sendo armazenados

Interrupt-driven I/O usando interrupção vetorizada

6: The ISR returns, thus restoring the PC to 100+1=101, where the µP resumes



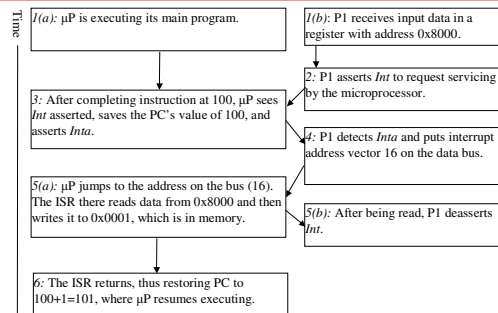
Acesso Direto à Memória

- Bufferização
 - Armazenamento temporário de dados em buffer antes de processamento
 - Dados acumulados em periféricos são bufferizadas
- Microprocessador pode tratar isto na ISR
 - Armazenamento e recuperação pelo processador é ineficiente
 - Programa deve esperar
- Usar Controlador de DMA é mais eficiente
 - Processador de propósito único separado
 - Processador libera controle do barramento ao DMA
 - Processador pode executar programa regularmente

Tabela de Endereços de Interrupção

- Compromisso entre interrupções fixas e vetorizadas
 - Um pino de interrupção
 - Tabela na memória armazenando endereços das ISRs
 - Periférico não fornece endereço e sim um valor de índice na tabela
 - Menos bits são enviados pelo periférico
 - Pode mudar ISR sem mudar periférico

Transferência periférico-memória sem DMA usando interrupção vetorizada



Transferência periférico-memória sem DMA usando interrupção vetorizada

1(a): μP is executing its main program

1(b): P1 receives input data in a register with address 0x8000.

Program memory
 ISR
 16: MOV R0, 0x8000
 17: # modifies R0
 18: MOV 0x0001, R0
 19: RETI # ISR return
 ...
 Main program
 ...
 100: instruction
 101: instruction

Data memory
 0x0000 0x0001

System bus

μP

Inta Int

P1
 16
 0x8000

PC
 100

Engenharia de Sistemas Embarcados 37

Transferência periférico-memória sem DMA usando interrupção vetorizada

4: P1 detects *Inta* and puts interrupt address vector 16 on the data bus.

Program memory
 ISR
 16: MOV R0, 0x8000
 17: # modifies R0
 18: MOV 0x0001, R0
 19: RETI # ISR return
 ...
 Main program
 ...
 100: instruction
 101: instruction

Data memory
 0x0000 0x0001

System bus

μP

Inta Int

P1
 16
 0x8000

PC
 100

Engenharia de Sistemas Embarcados 40

Transferência periférico-memória sem DMA usando interrupção vetorizada

2: P1 asserts *Int* to request servicing by the microprocessor

Program memory
 ISR
 16: MOV R0, 0x8000
 17: # modifies R0
 18: MOV 0x0001, R0
 19: RETI # ISR return
 ...
 Main program
 ...
 100: instruction
 101: instruction

Data memory
 0x0000 0x0001

System bus

μP

Inta Int

P1
 16
 0x8000

PC
 100

Engenharia de Sistemas Embarcados 38

Transferência periférico-memória sem DMA usando interrupção vetorizada

5(a): μP jumps to the address on the bus (16). The ISR there reads data from 0x8000 and then writes it to 0x0001, which is in memory.

5(b): After being read, P1 de-asserts *Int*.

Program memory
 ISR
 16: MOV R0, 0x8000
 17: # modifies R0
 18: MOV 0x8001, R0
 19: RETI # ISR return
 ...
 Main program
 ...
 100: instruction
 101: instruction

Data memory
 0x0000 0x0001

System bus

μP

Inta Int

P1
 16
 0x8000

PC
 100

Engenharia de Sistemas Embarcados 41

Transferência periférico-memória sem DMA usando interrupção vetorizada

3: After completing instruction at 100, μP sees *Int* asserted, saves the PC's value of 100, and asserts *Inta*.

Program memory
 ISR
 16: MOV R0, 0x8000
 17: # modifies R0
 18: MOV 0x0001, R0
 19: RETI # ISR return
 ...
 Main program
 ...
 100: instruction
 101: instruction

Data memory
 0x0000 0x0001

System bus

μP

Inta Int

P1
 16
 0x8000

PC
 100

Engenharia de Sistemas Embarcados 39

Transferência periférico-memória sem DMA usando interrupção vetorizada

6: The ISR returns, thus restoring PC to 100+1=101, where μP resumes executing.

Program memory
 ISR
 16: MOV R0, 0x8000
 17: # modifies R0
 18: MOV 0x0001, R0
 19: RETI # ISR return
 ...
 Main program
 ...
 100: instruction
 101: instruction

Data memory
 0x0000 0x0001

System bus

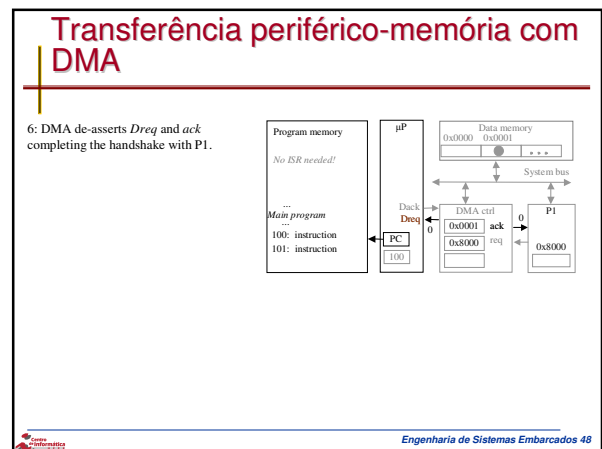
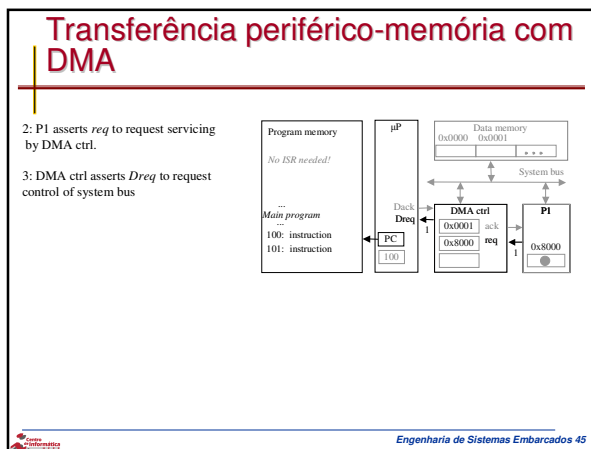
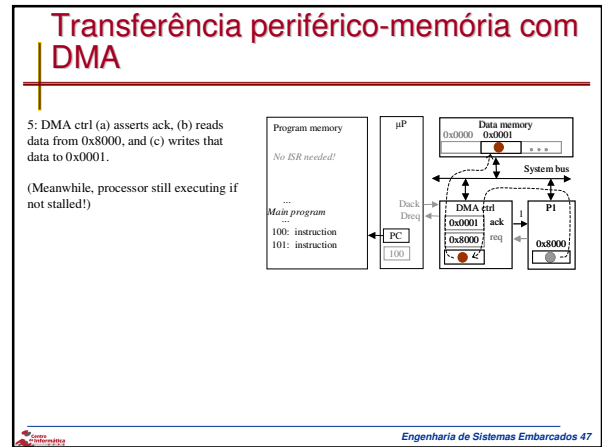
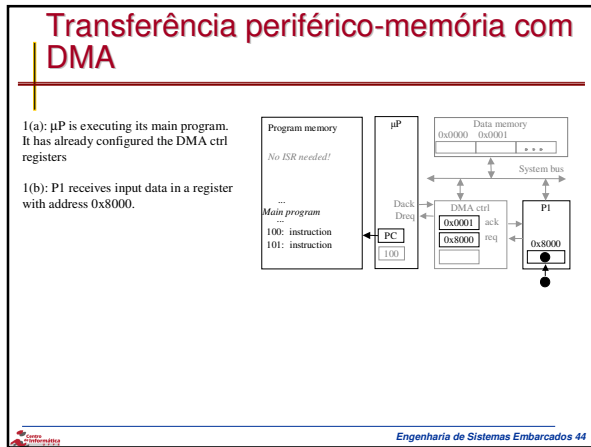
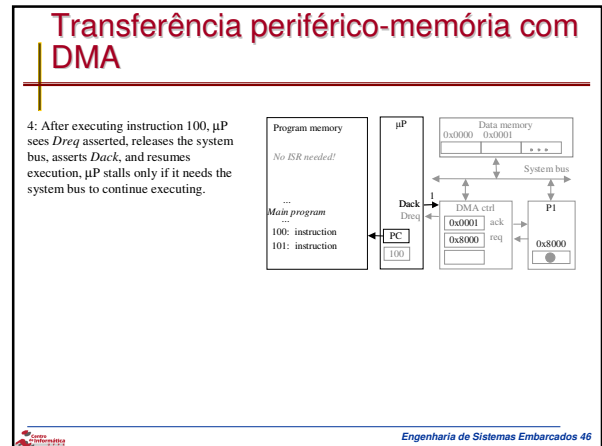
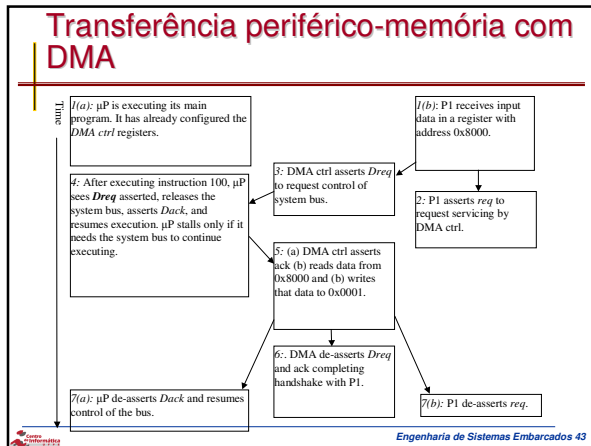
μP

Inta Int

P1
 16
 0x8000

PC
 101

Engenharia de Sistemas Embarcados 42



Ciclos de DMA em ISA bus

Engenharia de Sistemas Embarcados 49

Arbitragem através de Prioridades

- Tipos de Prioridades
 - Prioridade Fixa
 - Cada periférico tem prioridade fixa
 - Preferido quando existem periféricos de diferentes prioridades
 - Prioridade Rotativa (round-robin)
 - Prioridade muda em função do histórico de pedidos
 - Melhor distribuição de serviços entre periféricos com prioridades similares

Engenharia de Sistemas Embarcados 52

Arbitragem através de Prioridades

- Considere a situação onde vários periféricos solicitam simultaneamente serviços de um único recurso (processador, DMA,...). Qual deve ser atendido primeiro?
- Árbitro de Prioridade
 - Processador de propósito único
 - Periféricos requisitam ao árbitro e árbitro requisita o recurso

Engenharia de Sistemas Embarcados 50

Arbitragem: Daisy-chain

- Arbitragem feita pelos periféricos
 - Interna no periférico ou por lógica adicional
 - Entrada req e saída ack adicionada a cada periférico
- Periféricos conectados em cadeia (daisy-chain)
 - Requisição flui iniciando do periférico mais próximo ao recurso (alta prioridade)

Engenharia de Sistemas Embarcados 53

Arbitragem através de Prioridades

1. Microprocessor is executing its program.
2. Peripheral1 needs servicing so asserts *Ireq1*. Peripheral2 also needs servicing so asserts *Ireq2*.
3. Priority arbiter sees at least one *Ireq* input asserted, so asserts *Int*.
4. Microprocessor stops executing its program and stores its state.
5. Microprocessor asserts *Inta*.
6. Priority arbiter asserts *Iack1* to acknowledge Peripheral1.
7. Peripheral1 puts its interrupt address vector on the system bus
8. Microprocessor jumps to the address of ISR read from data bus, ISR executes and returns (and completes handshake with arbiter).
9. Microprocessor resumes executing its program.

Engenharia de Sistemas Embarcados 51

Arbitragem: Daisy-chain

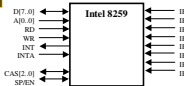
- Vantagens e desvantagens
 - Facilidade de incluir e remover periférico
 - Não suporta prioridade rotativa
 - Defeito em periférico pode inviabilizar sistema

Engenharia de Sistemas Embarcados 54

Arbitragem orientada a Barramento

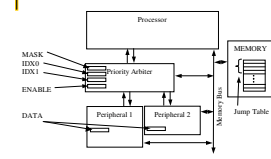
- Quando vários processadores compartilham o barramento
 - Arbitragem embutida no protocolo de barramento
 - Processadores podem tentar escrever simultaneamente causando colisões:
 - Data deve ser reenviado
 - Só um inicia reenvio

Intel 8259 programmable priority controller



Signal	Description
DT[7:0]	These wires are connected to the system bus and are used by the microprocessor to write or read the internal registers of the 8259.
A[0:0]	This pin acts in conjunction with WR/RD signals. It is used by the 8259 to decipher various command words the microprocessor writes and status the microprocessor wishes to read.
WR	When this write signal is asserted, the 8259 accepts the command on the data bus, i.e., the microprocessor writes to the 8259 by asserting this signal.
RD	When this read signal is asserted, the 8259 provides on the data bus its status, i.e., the microprocessor reads the status of the 8259 by asserting this signal and reading the data bus.
INT	This signal is asserted whenever a valid interrupt request is received by the 8259, i.e., it is used to interrupt the microprocessor.
INTA	This signal is used to enable 8259 interrupt-sector data onto the data bus by a sequence of interrupt acknowledge pulses issued by the microprocessor.
IR[1,2,3,4,5,6,7]	An interrupt request is executed by a peripheral device when one of these signals is asserted.
CAS[2:0]	These are cascade signals to enable multiple 8259 chips to be chained together.
SPKIN	This function is used in conjunction with the CAS signals for cascading purposes.

Exemplo: Interrupção Vetorizada usando Tabela de Interrupção



- Prioridade Fixa: periférico tem mais alta prioridade
- Keyword "_at_" seguida a endereço de memória força compilar a armazenar variáveis em endereços específicos
- Índice dos periféricos na tabela é enviado ao arbiter
- Periféricos recebem dados e solicitam interrupção

```

unsigned char ARBITER_MASK_REG      _at_ 0x00;
unsigned char ARBITER_CH0_INDEX_REG _at_ 0x01;
unsigned char ARBITER_CH1_INDEX_REG _at_ 0x02;
unsigned char ARBITER_ENABLE_REG    _at_ 0x03;
unsigned char PERIPHERAL1_DATA_REG  _at_ 0x04;
unsigned char PERIPHERAL2_DATA_REG  _at_ 0x05;
unsigned void* INTERRUPT_VECTOR_TABLE[256] _at_ 0x06;

void main() {
    initializePeripherals();
    for(;;) {} // main program goes here
}

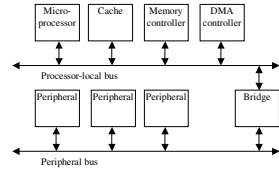
void Peripheral1_ISR(void) {
    unsigned char data;
    data = PERIPHERAL1_DATA_REG;
    // do something with the data
}

void Peripheral2_ISR(void) {
    unsigned char data;
    data = PERIPHERAL2_DATA_REG;
    // do something with the data
}

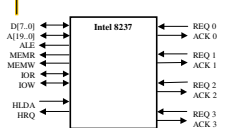
void InitializePeripherals(void) {
    ARBITER_MASK_REG = 0x03; // enable both channels
    ARBITER_CH0_INDEX_REG = 1;
    ARBITER_CH1_INDEX_REG = 17;
    INTERRUPT_VECTOR_TABLE[1] = (void*)Peripheral1_ISR;
    INTERRUPT_VECTOR_TABLE[17] = (void*)Peripheral2_ISR;
    ARBITER_ENABLE_REG = 1;
}
    
```

Barramentos Multi-níveis

- Barramento local ao Processador
 - Alta velocidade e largura para comunicações frequentes
 - Conecta processador, cache, e controladores de memória
- Barramento de Periféricos
 - Baixa velocidade, mais estreitos, comunicações menos frequentes
 - Padrões industriais (ISA, PCI)



Intel 8237 DMA controller



Signal	Description
DT[7:0]	These wires are connected to the system bus (ISA) and are used by the microprocessor to write to the internal registers of the 8237.
A[19:0]	These wires are connected to the system bus (ISA) and are used by the DMA to issue the memory location where the transferred data is to be written to. The 8237 &
ALE*	This is the address latch enable signal. The 8237 use this signal when driving the system bus (ISA).
MEMR*	This is the memory read signal issued by the 8237 when driving the system bus (ISA).
MEMW*	This is the memory read signal issued by the 8237 when driving the system bus (ISA).
IOR*	This is the IO device read signal issued by the 8237 when driving the system bus (ISA) in order to read a byte from an IO device.
IOW*	This is the IO device write signal issued by the 8237 when driving the system bus (ISA) in order to write a byte to an IO device.
HLDA	This signal (hold acknowledge) is asserted by the microprocessor to signal that it has relinquished the system bus (ISA).
HRQ	This signal (hold request) is asserted by the 8237 to signal to the microprocessor a request to relinquish the system bus (ISA).
REQ 0,1,2,3	An attached device to one of these channels asserts this signal to request a DMA transfer.
ACK 0,1,2,3	The 8237 asserts this signal to grant a DMA transfer to an attached device to one of these channels.

Comunicação avançada

- Estruturação em Camadas
 - Complexidade dos protocolos de comunicação é distribuídas em camadas mais fáceis de entender e projetar
 - Camadas em níveis inferiores oferecem serviços para camadas em níveis superiores
 - Nível Físico
 - Nível mais baixo na hierarquia
 - Meio para transferência de dados entre dispositivos
- Comunicação Paralela
 - Meio físico capaz de transportar múltiplos bits
- Comunicação Serial
 - Meio físico capaz de transportar um bit por vez
- Comunicação Wireless
 - Não há a necessidade de meio físico para transporte da informação

Comunicação Paralela

- Múltiplos dados, controle e fios de potência
 - Um bit por fio
- Altas taxas de transferências e distâncias curtas
- Usado na conexão de dispositivos no mesmo IC ou placa
 - Deve ser pequeno
 - Fios mais longos exigem mais tempo de transmissão
- Alto custo

Detecção de Erro e Correção

- Parte do protocolo de barramento
- Detecção de erro: habilidade de receptor detectar erros durante transmissão
- Correção de erro: habilidade do transmissor e receptor cooperarem para corrigir o problema
 - Protocolo reconhecimento/retransmissão
- Erro de Bit : único bit é invertido
- Burst de erros: bits consecutivos são recebidos de forma incorreta
- Paridade: bit extra enviado com palavra para detecção do erro
 - Paridade Par: dado mais bit de paridade resultam número par de 1's
 - Paridade Ímpar: dado mais bit de paridade resultam número ímpar de 1's
 - Sempre detecta erros em 1 bit mas não em mais de 1 bit
- Checksum: palavra extra enviada como dado para múltiplas palavras
 - e.x., palavra extra contem soma XOR de todas as palavras no pacote

Comunicação Serial

- Único fio para dado, controle e tensão de alimentação
- Palavras transmitidas bit a bit
- Altas taxas com distâncias longas
 - Mais bits por unidade de tempo
- Mais baratos e com menos interferência
- Lógica de Interface e Protocolo mais complexos
 - Transmissor precisa decompor palavras
 - Receptor precisa recompor palavras
 - Sinais de protocolo enviados no mesmo fio torna protocolo mais complexo

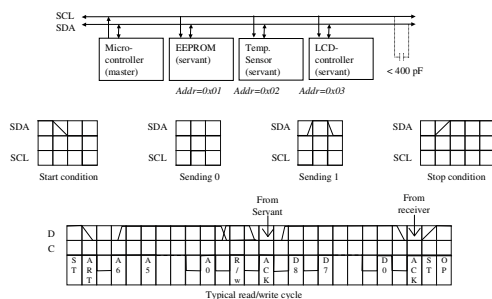
Protocolos Seriais : I²C

- I²C (Inter-IC)
 - Protocolo serial para barramento de dois fios desenvolvido pela Philips a 20 anos
 - Permite comunicação entre dispositivo periférico de forma simples
 - Taxa de transferência de 100 kbits/s e endereços de 7-bits em modo normal
 - 3.4 Mbits/s e endereços de 10-bits no modo rápido
 - Dispositivos que possuem interface com barramento I²C:
 - EPROMs, Flash, e algumas memórias RAM, clocks de tempo real, watchdog timers, e microcontroladores

Comunicação Wireless

- Infra-vermelho (IR)
 - Frequência de ondas eletrônicas abaixo do espectro da luz visível
 - Diodo emite luz infra-vermelha para gerar sinal
 - Transistor detecta sinal e conduz
 - Produção Barata
 - Precisa linha de visagem , range limitado
- Radio frequência (RF)
 - Frequências de Ondas eletromagnéticas no espectro de rádio
 - Circuito análogo e antenas são necessários para transmissão e recepção
 - Linha de visagem não é necessária, potência de transmissor determina range

Estrutura de Barramento I²C



Protocolos Seriais: CAN

- CAN (Controller area network)
 - Protocolo para aplicações de tempo real
 - Desenvolvido por Robert Bosch GmbH
 - Originalmente para comunicação entre dispositivos na eletrônica automotiva
 - Aplicações atuais incluem:
 - Controladores de elevador, copiadores, telescópios, sistemas de controle de linha de produção e instrumentos médicos
 - Taxa de Transferência de 1 Mbit/s e endereços de 11-bits
 - Dispositivos que possuem interface CAN:
 - Processadores compatíveis com 8051 e controladores CAN



Engenharia de Sistemas Embarcados 67

Protocolos Paralelos: Barramento PCI

- Barramento PCI (Peripheral Component Interconnect)
 - Barramento de alto desempenho desenvolvido pela Intel no início dos anos 90
 - Padrão adotado pela indústria e administrado pela PCISIG (PCI Special Interest Group)
 - Interconecta chips, placas de expansão, subsistemas processador memória
 - Taxas de Transferência de 127.2 a 508.6 Mbits/s e endereços de 32-bits
 - Estendido para 64 bits
 - Barramento síncrono
 - Multiplexa linhas de endereço e dado



Engenharia de Sistemas Embarcados 70

Protocolos Seriais: FireWire

- FireWire (a.k.a. I-Link, Lynx, IEEE 1394)
 - Barramento serial de alto desempenho desenvolvido pela Apple Computer Inc.
 - Projetado para interface entre componentes eletrônicos
 - Taxa de Transferência de 12.5 a 400 Mbits/s, e endereços de 64-bits
 - Capacidade Plug-and-play
 - Estruturação em camadas
 - Aplicações que possuem interface FireWire incluem:
 - drives, impressoras, scanners, câmeras
 - Capaz de suportar LAN (similar a Ethernet)
 - Endereços de 64-bits:
 - 10 bits para ids de rede (1023 subredes)
 - 6 bits para node ids (cada subrede pode ter 64 nós)
 - 48 bits para endereço de memória (cada nó pode ter 281 terabytes)



Engenharia de Sistemas Embarcados 68

Protocolos Paralelos: ARM Bus

- ARM Bus
 - Projetado e usado pela ARM Corporation
 - Permite conexão de processadores da linha ARM
 - Muitas companhias possuem protocolo de barramento proprietário
 - Taxa de Transferência é função do clock
 - Se frequência do clock é X, taxa de transferência é = 16 x X bits/s
 - Endereços de 32-bits



Engenharia de Sistemas Embarcados 71

Protocolos Seriais: USB

- USB (Universal Serial Bus)
 - Facilita conexão entre PC e monitores, impressoras, modems, scanners, e dispositivos multimídia
 - 2 taxas de transferências:
 - 12 Mbps para dispositivos rápidos
 - 1.5 Mbps para dispositivos mais lentos
 - Topologia em estrela pode ser usada
 - Um dispositivo USB (hub) conectado ao PC
 - Múltiplos dispositivos USB podem ser conectados ao hub
 - Até 127 podem ser conectados desta forma
 - Controlador host USB
 - Gerencia e controla taxas e parâmetros de software requeridos por cada periférico
 - Gerencia potência dinamicamente em função dos dispositivos conectados



Engenharia de Sistemas Embarcados 69

Protocolos Wireless: IrDA

- IrDA
 - Protocolo que suporta transmissão em pequeno range ponto a ponto usando infra-vermelho
 - Criado e promovido pela Infrared Data Association (IrDA)
 - Taxa de Transferência de 9.6 kbps e 4 Mbps
 - IrDA hardware disponível em notebooks, impressoras, PDAs, câmeras digitais, telefones celulares
 - Falta de drivers tem prejudicado desenvolvimento de aplicações
 - Disponível em S.O embarcados



Engenharia de Sistemas Embarcados 72

Protocolos Wireless: Bluetooth

- Bluetooth
 - Novidade, padrão para conectividade sem fio
 - Baseado em link de rádio de curto alcance e baixo custo
 - Conexão estabelecida dentro de 10 metros
 - Nenhuma linha de sight requerida
 - e.x., Conecta com impressora em outra sala



Engenharia de Sistemas Embarcados 73

Protocolo Wireless: IEEE 802.11

- IEEE 802.11
 - Proposto como padrão para LANs sem fio
 - Especifica parâmetros para níveis PHY e MAC da rede
 - Camada PHY
 - Camada física
 - Transmissão de dados entre nós
 - Taxa de transferência de 1 ou 2 Mbps
 - Opera na frequência de 2.4 to 2.4835 GHz (RF)
 - ou 300 a 428,000 GHz (IR)
 - Camada MAC
 - Camada de controle de acesso ao meio
 - Protocolo gerencia uso de recurso compartilhado
 - Protocolo baseado em colisão



Engenharia de Sistemas Embarcados 74

Resumo

- Conceitos básicos de protocolos
 - Atores, direção, multiplexação no tempo, controle
- Processadores de Propósito Geral
 - E/S baseada em portas ou em barramento
 - Endereçamento de E/S: Mapeada em Memória ou E/S Padrão
 - Tratamento de Interrupções: fixa ou vetorizada
 - DMA
- Arbitragem
 - Arbitro com Prioridades (fixa ou rotativa) ou daisy chain
- Hierarquia de Barramentos
- Comunicações Avançadas
 - Paralela vs. serial, com fios vs. wireless, detecção e correção de erros, camadas
 - Protocolos seriais: I²C, CAN, FireWire, e USB; Paralelos: PCI e ARM.
 - Protocolos seriais wireless : IrDA, Bluetooth, e IEEE 802.11.



Engenharia de Sistemas Embarcados 75