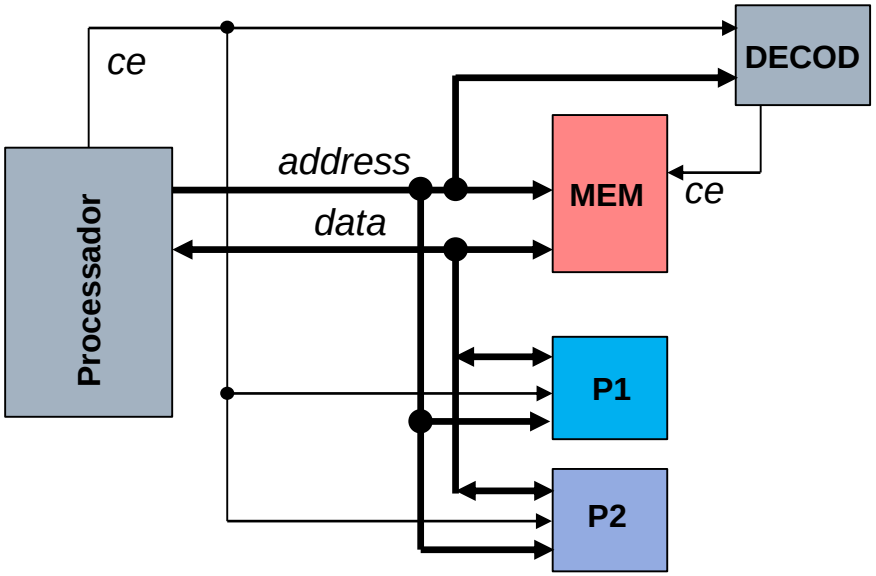
Interface entre Processador e Periféricos

- Transferência de dados
  - Existem basicamente três técnicas envolvidas na transferência de dados entre processador e periféricos
    - *Polling*
    - Interrupção
    - Acesso direto à memória (DMA – *Direct Memory Access*)



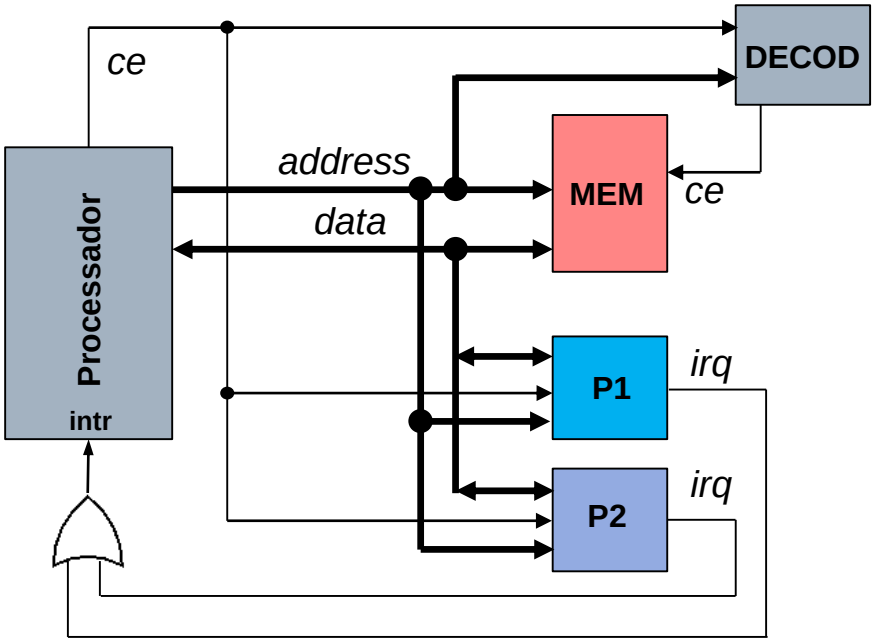
Interface entre Processador e Periféricos

- Interrupção
  - Criada para solucionar o problema do tempo desperdiçado com a verificação de estado do periférico inerente ao *polling*
  - O periférico é responsável por "chamar a atenção" do processador
    - Existem dados para serem lidos
    - Periférico está pronto para receber dados
    - "Lembrar" o processador que alguma tarefa periódica deve ser executada
  - Em seguida o processador atende ao pedido de interrupção e executa a rotina de tratamento de interrupção (*ISR – Interrupt Service Routine*)
    - Trecho de código que verifica o periférico origem da interrupção e desvia para o manipulador correspondente (*handler*)
    - Para cada perférico existe um *handler*
      - Código que interage com o periférico (*device driver*)

Interface entre Processador e Periféricos

Interrupção

- O processador possui uma entrada de interrupção (e.g. *intr*), a qual é utilizada pelos periféricos para "chamar a atenção"
- A interrupção é gerada pelos periféricos através de um bit de interrupção (*irq – interrupt request*)
- Interrupções podem ocorrer a qualquer instante (assíncronas)

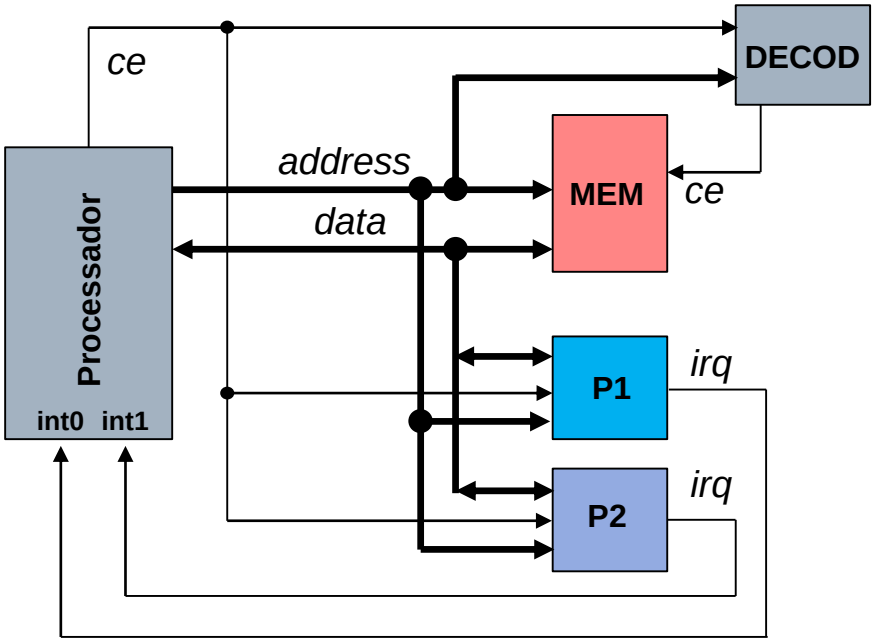


Interface entre Processador e Periféricos

Interrupção

- O processador possui uma entrada de interrupção (e.g. *intr*), a qual é utilizada pelos periféricos para "chamar a atenção"
- A interrupção é gerada pelos periféricos através de um bit de interrupção (*irq – interrupt request*)
- Interrupções podem ocorrer a qualquer instante (assíncronas)

Pode existir mais de uma entrada de interrupção no processador

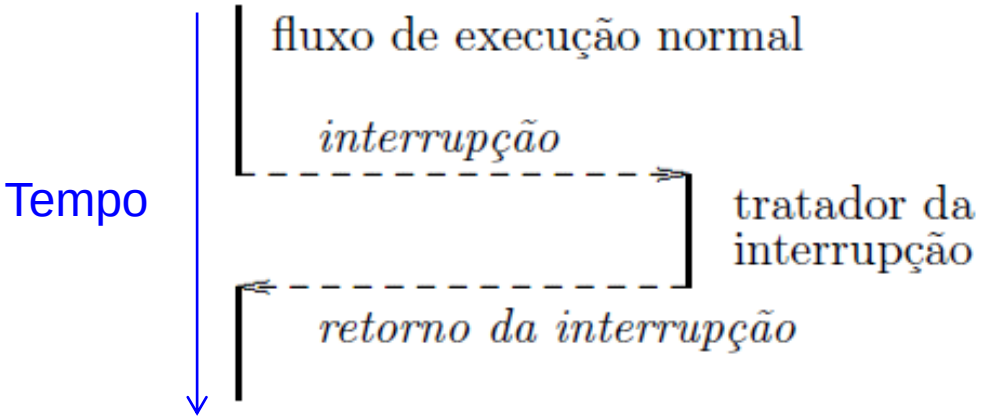


Interface entre Processador e Periféricos

- Interrupção
  - No caso do *CryptoMessage*, ao invés de fazer *polling* no sinal *keyExchange*, ele poderia chamar a atenção do processador sempre que um dado estiver disponível
    - *Processador.intr* ← *CryptoMessage.keyExchange*
  - Enquanto o *CriptoMessage* não tem mensagem, o processador pode executar outro programa
  - Dessa maneira, o processador divide sua "atenção" entre periféricos e programas
    - De tempos em tempos o processador atente o periférico quando este gera uma interrupção
    - Caso contrário algum outro programa é executado
    - Fundamento de sistemas multi-tarefa
    - A execução intercalada de programas produz uma ilusão de paralelismo para o usuário

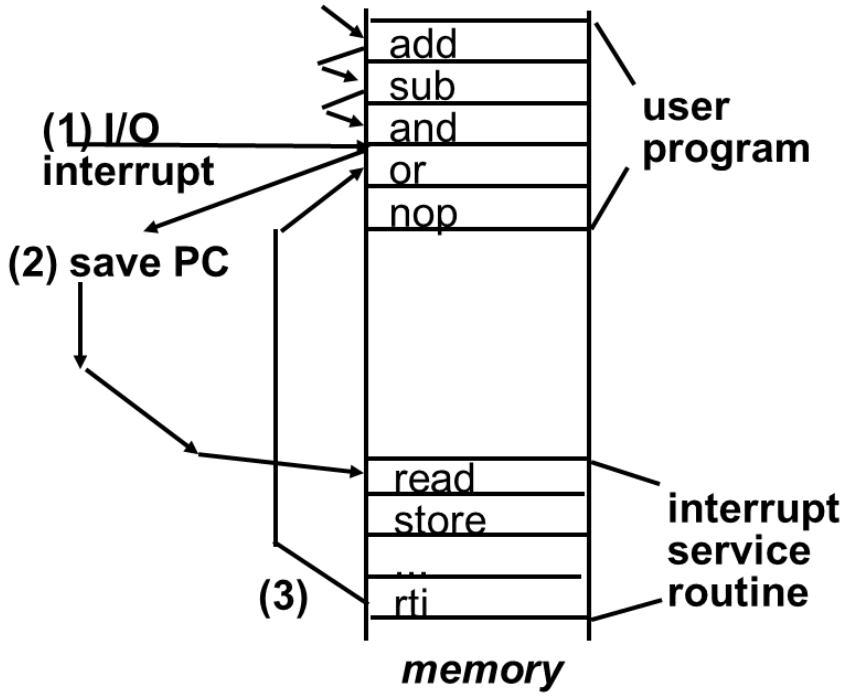
Interface entre Processador e Periféricos

- Interrupção
  - Este mecanismo se chama de interrupcão porque a sequência normal de execução de um programa é interrompida para que um periférico seja atendido
  - O tratamento de uma interrupção é similar a uma chamada de função
  - Ao final do tratamento da interrupção, o processador retorna ao fluxo de intruções que era executado antes da interrupção



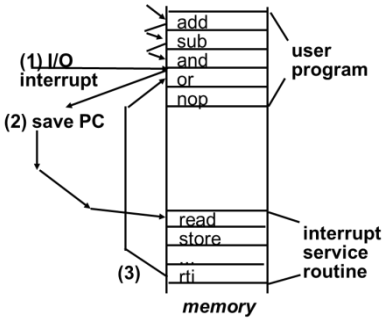
Interface entre Processador e Periféricos

- Interrupção
  - Ao invés de o programador inserir no código uma chamada para a ISR (e.g. *jump*), é o periférico que ao interromper dispara a sua execução



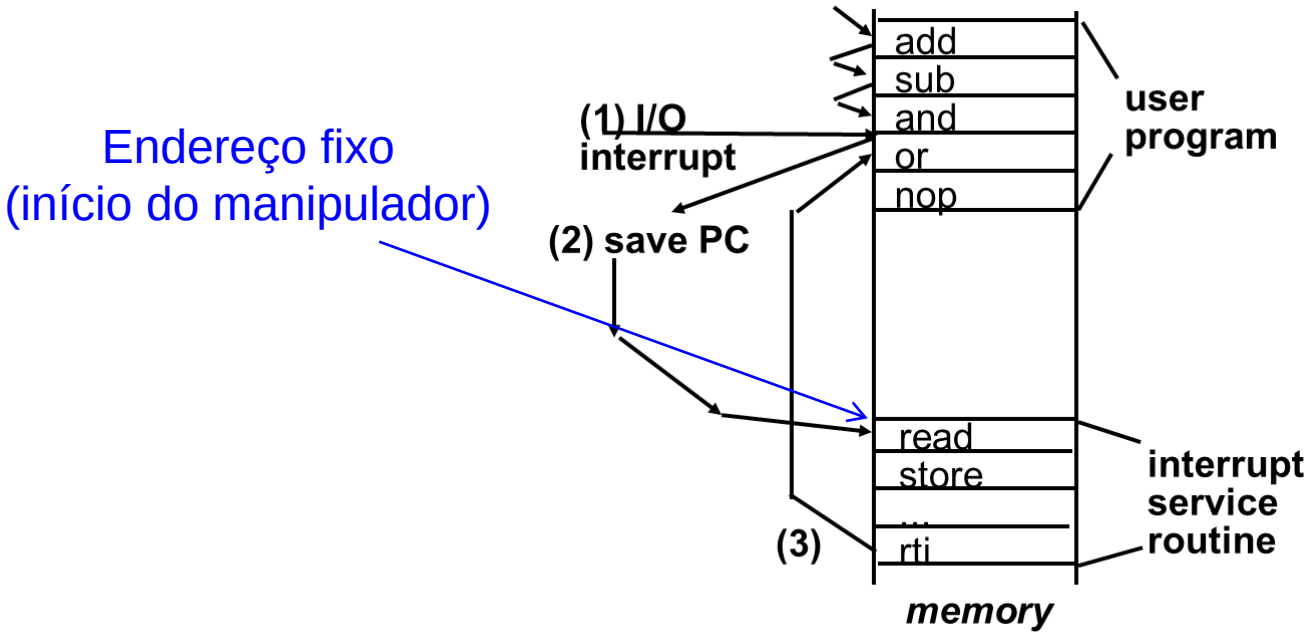
Interface entre Processador e Periféricos

- Interrupção (tratamento)
  - Antes de buscar a próxima instrução do programa em execução, o processador verifica se há um pedido de interrupção
    - Se não há nenhum pedido de interrupção, a próxima instrução do programa é buscada
    - Se há um pedido de interrupção, o processador interrompe a sequência normal de execucão das instruções do programa e desvia para a ISR
      - O endereço da instrução que seria executada é armazenado e o PC é setado com o endereço do manipulador
      - A primeira instrução da ISR é buscada, inciando sua excução
      - Ao final do tratamento da interrupção, o processador salta de volta para o endereço da instrução do programa que não foi executada por conta da interrupção (recupera o PC salvo)



Interface entre Processador e Periféricos

- Interrupção
  - O endereço de desvio quando ocorre uma interrupção é tipicamente fixo (*hardwired*) e varia de processador para processador
  - Portanto, o código da ISR deve sempre ser alocado na mesma posição na memória



Interface entre Processador e Periféricos

- Interrupção
  - Enquanto a técnica *polling* é puramente implementada em software, a E/S com interrupção requer um suporte de hardware
  - A vantagem é que o processador não precisa ficar verificando constantemente o estado do periférico
  - Entretanto, a transferência de dados continua sendo realizada pelo processador
    - Para liberar o processador da transferência de dados entre periférico e memória pode-se utilizar um circuito DMA

Interface entre Processador e Periféricos

- Trabalho 3 – parte 1
  - Adicionar ao processador R8 capacidade de atender interrupções
  - Sempre que um pedido de interrupção for feito, o processador deve desviar para uma ISR e executar a rotina de tratamento
    - Se o pedido de interrupção ocorrer no meio da execução de uma instrução, a instrução deve terminar a execução antes de desviar para a ISR
  - Em seguida a execução deve retornar de onde foi interrompida
  - Na interface do processador deve ser adicionado uma entrada de interrupção *intr* (*interrput request*)

Interface entre Processador e Periféricos

- Trabalho 3 – parte 1
  - A entrada *intr* deve ser registrada a fim de não perder pedidos de interrupção curtos
    - Exemplo: Ocorre um pulso de 1 ciclos de *clock* na entrada *intr* no meio da execução de um instrução de 4 ciclos
    - Considerar que a entrada *intr* deve ficar ativa pelo menos 1 ciclo de *clock* para que o pedido de interrupção seja considerado
  - O desvio para a ISR é como uma chamada de subrotina
    - Dica: tomar como base a execução da instrução JSR
  - O manipulador de interrupções deve ser alocado a partir de um endereço fixo na memória
  - Portanto ao ocorrer uma interrupção, o PC recebe este endereço (salto implícito)

Interface entre Processador e Periféricos

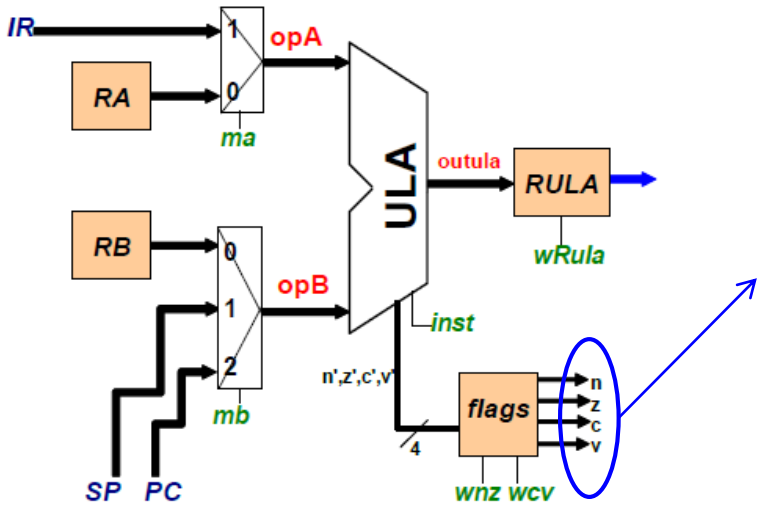
- Trabalho 3 – parte 1
  - Duranto o atendimento de um pedido de interrupção, o processador deve ignorar novos pedidos, de maneira a não ser interrompido novamente (ISR não reentrante)
  - O retorno de uma interrupção é semelhante ao retorno de uma subrotina
  - Para tanto, deve ser adicionada a instrução RTI (*Return from interruption*)
    - Definir um *opcode* não utilizado pelas demais instruções da arquitetura
    - Dica: tomar com base a instrução RTS
    - Após executar RTI o processador pode tratar novos pedidos de interrupção

Interface entre Processador e Periféricos

- Trabalho 3 – parte 1
  - Salvamento de contexto
    - Ao iniciar a rotina de tratamento da interrupção, a primeira coisa a ser feita é armazenar na pilha todos registradores utilizados por esta rotina e os *flags*
    - O objetivo é manter consistente os registradores utilizados pelo programa interrompido
  - Recuperação do contexto
    - Imediatamente antes do retorno da interrupção (RTI) o conteúdo dos registradores e os *flags* devem ser restaurados

Interface entre Processador e Periféricos

- Trabalho 3 – parte 1
  - Adicionar ao processador R8 instruções para armazenar/rescuperar da pilha os *flags*
    - PUSHF: Armazena *flags* na pilha
    - POPF: Recupera *flags* da pilha
    - Processadores da Intel (x86) tem essas instruções
    - Dica: tomar como base as instruções PUSH e POP

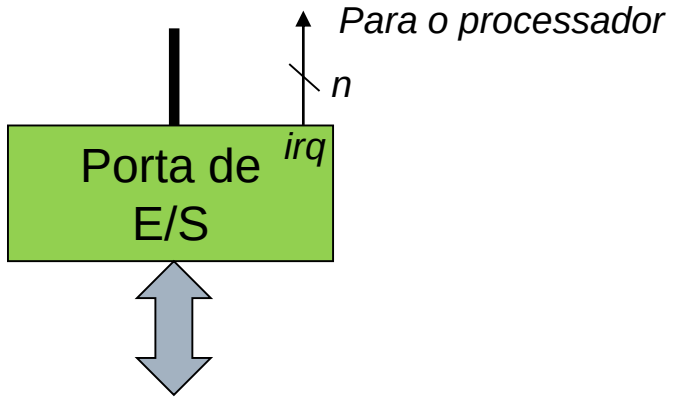


Apresentar FSM atualizada

Juntar em uma palavra de 16 bits para armazenar na pilha
Exemplo: x"000" & n & z & c & v

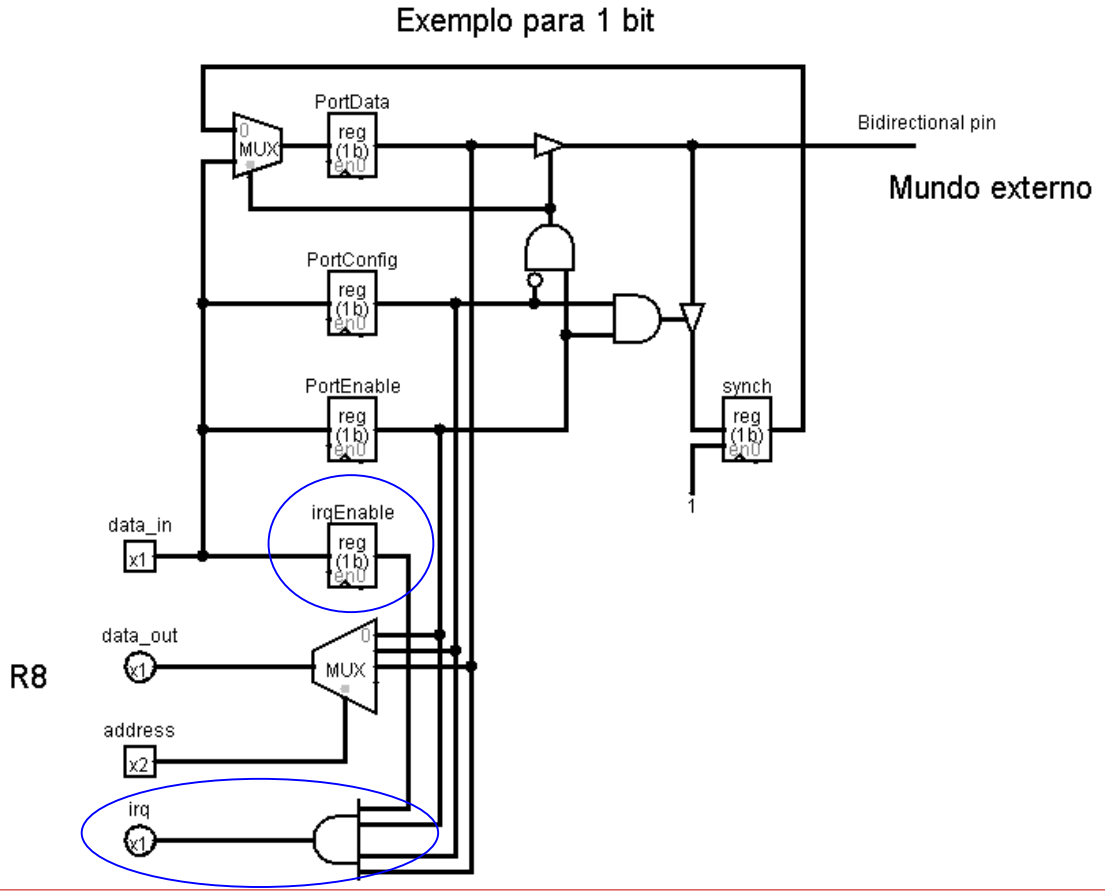
Interface entre Processador e Periféricos

- Trabalho 3 – parte 1
  - Adicionar saída de interrupção (*irq*) na porta de E/S
    - A saída *irq* deve ter a mesma largura da porta de E/S
    - Qualquer bit da porta de E/S pode ser configurado como entrada de interrupção
    - Deve ser adicionado outro registrador à porta de E/S a fim de configurar entradas de interrupção
    - Os bits da porta de E/S configurados como entrada de interrupção são ligados aos bits correspondentes da saída *irq*
      - *Exemplo: bits 1 e 3 da porta de E/S configurados como entadas de interrupção*
        - *irq(1) ← port_io(1)*
        - *irq(3) ← port_io(3)*



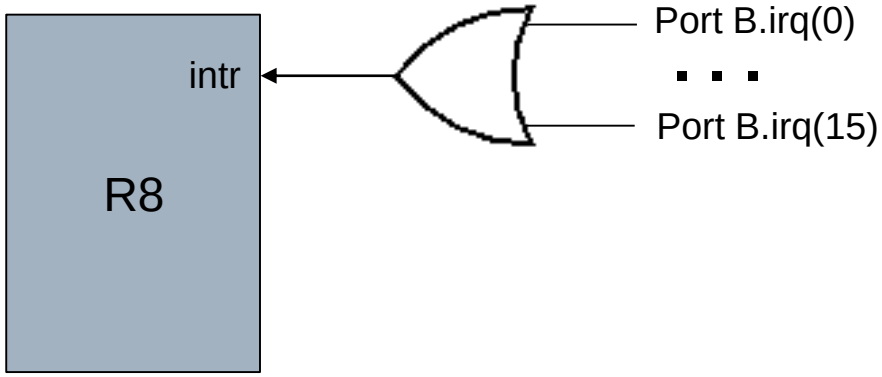
Interface entre Processador e Periféricos

- Trabalho 3 – parte 1
  - Adicionar saída de interrupção (*irq*) na porta de E/S



Interface entre Processador e Periféricos

- Trabalho 3 – parte 1
  - A entrada de interrupção do processador deve receber o *or* de todas saídas *irq* da Porta B



Interface entre Processador e Periféricos

- Trabalho 3 – parte 1
  - Aplicação
    - Configurar o *keyExchange* dos *CryptoMessages* como entradas de interrupção
    - A aplicação principal executada pelo processador será o *bubbleSort*
      - Aumentar o tamanho do *array* para 50 elementos
    - Ao ser interrompido, o processador deve verificar a origem da interrupção lendo as entradas de interrupção e verificando a que está ativa (*polled interrupt*)
      - O ordem de leitura determina a prioridade no atendimento dos periféricos
      - Ao detectar o periférico, saltar para o *handler* correspondente
      - Configurar o tempo ocioso dos *CryptoMessages* para que eles interrompam o processador pelo menos 5 vezes cada durante a execução do *bubbleSort*

Interface entre Processador e Periféricos

- Trabalho 3 – parte 1
  - Estrutura da ISR

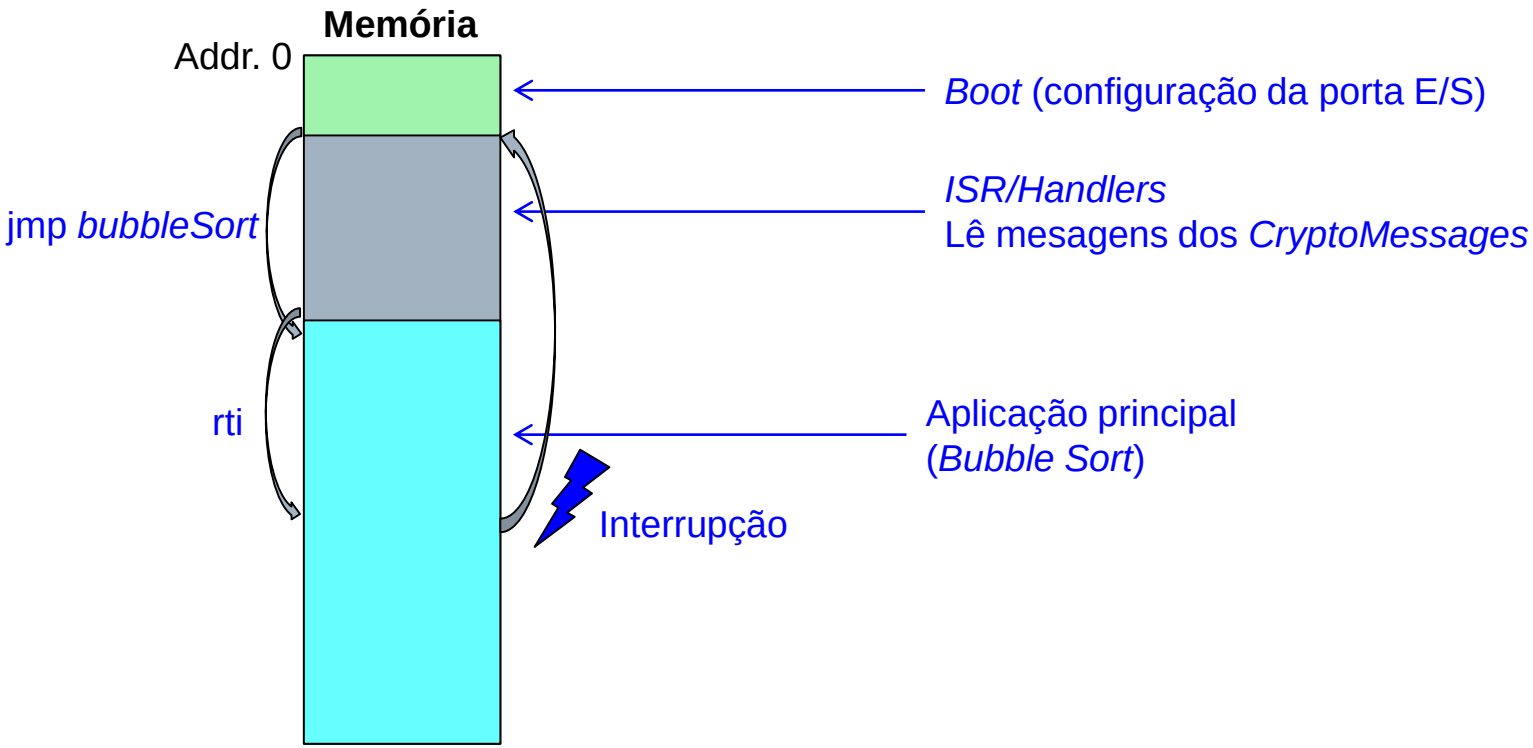
```
InterruptionServiceRoutine:
    1. Salvamento de contexto
    2. Verificação da origem da interrupção e salto para
       handler correspondente (jsr)
    3. Recuperação de contexto
    4. Retorno (rti)

CriptoMessage1_Handler:
    ...
    rts

CriptoMessage2_Handler:
    ...
    rts
```

Interface entre Processador e Periféricos

- Trabalho 3 – parte 1
  - Estrutura do código na memória



Interface entre Processador e Periféricos

- Trabalho 3 – parte 1
  - Manter mesmos grupos dos trabalhos anteriores
  - Apresentação dia 12/5
  - A nota do trabalho dará ENORME ÊNFASE à execução correta da simulação
  - A apresentação será oral, teórico-prática, frente ao computador, onde o grupo deverá explicar ao professor o projeto, a simulação e a implementação