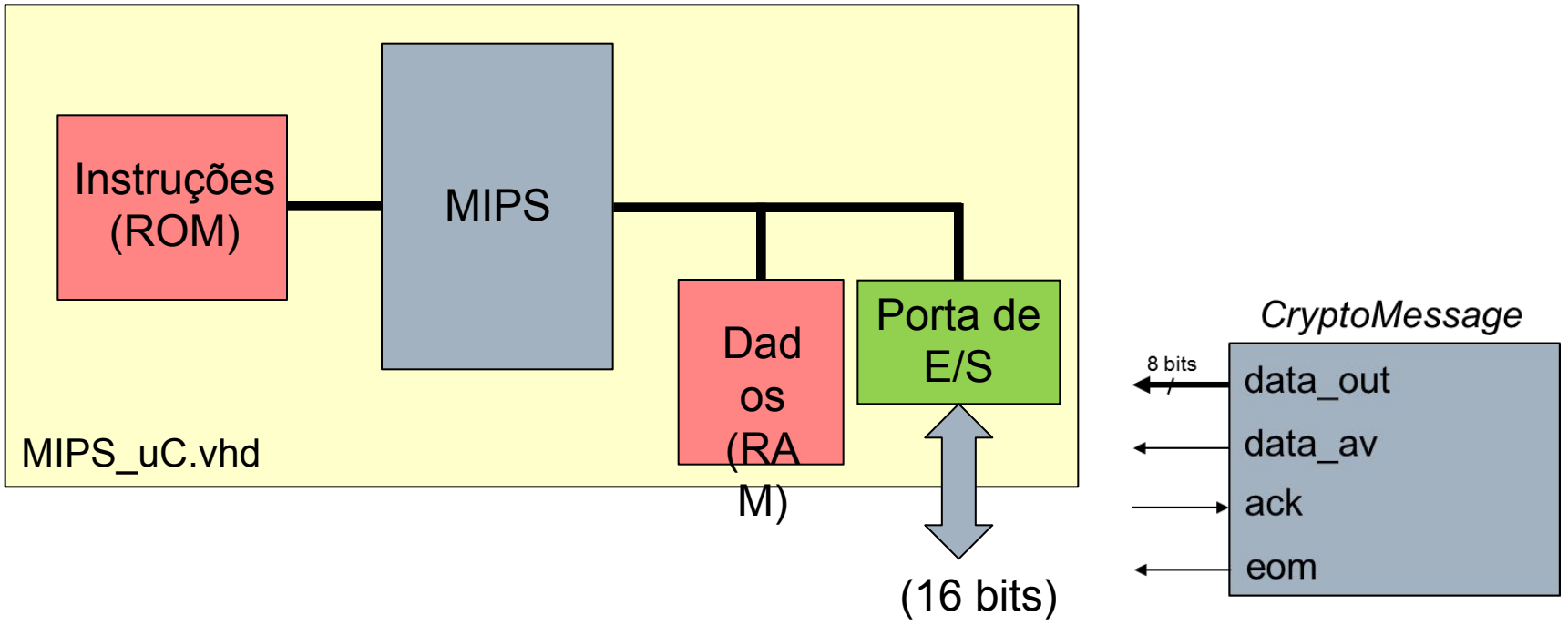
Interface entre Processador e Periféricos

Sub-sistema de entrada e saída – E/S (I/O)

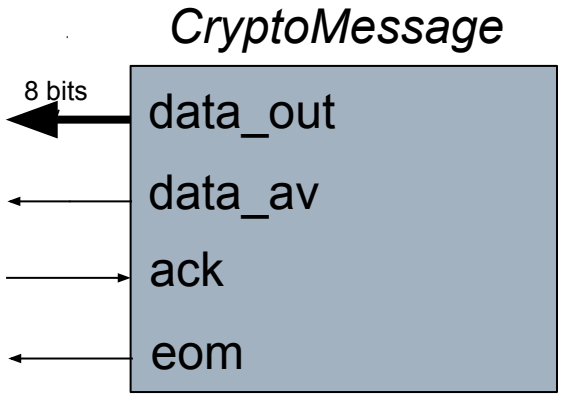
Interface entre Processador e Periféricos

- Trabalho 4 – parte 1
  - O objetivo do trabalho será conectar um periférico externo ao MIPS_uC através da porta de E/S



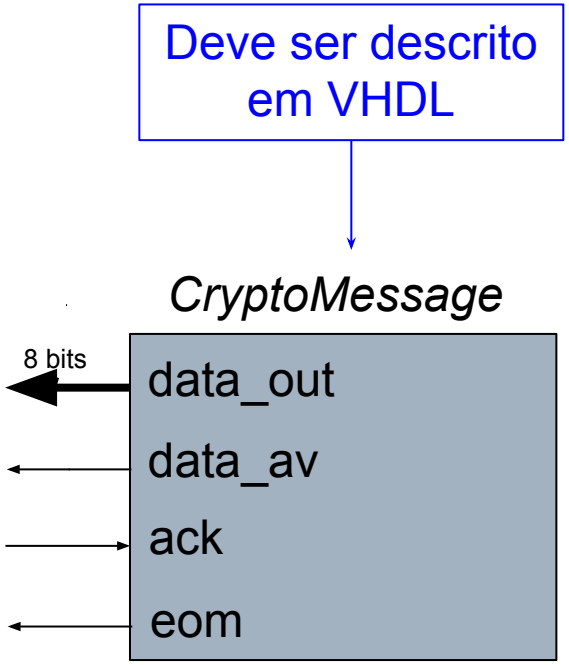
Interface entre Processador e Periféricos

- Trabalho 4 – parte 1
  - *CryptoMessage*
    - Envia mensagens criptografadas lidas de uma arquivo texto
    - Implementar em VHDL (não sintetizável)



Interface entre Processador e Periféricos

- Trabalho 4 – parte 1
  - *CryptoMessage*
    - Interação com o MIPS_uC



1. *CryptoMessage* coloca um byte da mensagem criptografada no barramento *data_out* e seta *data_av = 1*
2. Ao detectar *data_av = 1*, *MIPS_uC* lê o byte e seta *ack = 1*
3. Ao detectar *ack = 1*, *CryptoMessage* seta *data_av = 0*
4. Ao detectar data_av = 0, *MIPS_uC* seta *ack = 0*
5. Ao detectar *ack = 0*, *CryptoMessage* volta ao passo 1

Four-phase handshake protocol

Os passos de 1 a 5 se repetem até o final da mensagem. Uma mensagem corresponde a uma linha do arquivo texto.
Quando o *CryptoMessage* colocar o último byte da mensagem criptografada em *data_out*, *eom* (*end-of-message*) é ativado/desativado junto com *data_av*

Interface entre Processador e Periféricos

- Trabalho 4 – parte 1
  - Encriptação/Decriptação
    - rsa32_demo.c (*moodle*)
      - Versão com tamanho de chave reduzida (32 bits) do algoritmo de encriptação RSA
        - Tamanhos práticos de chaves são 1024, 2048, 3072, 4096, … bits
      - A parte do código relativa à encriptação deve ser implementada no *CryptoMessage* em VHDL
        - Tomar como base o CryptoMessage.vhd no *moodle* (leitura de arquivo, *ExpMod*, etc)
      - O *CryptoMessage* será usado apenas em simulação (não será sintetizado)
      - A parte do código relativa à decriptação deve ser implementada em *assembly*
      - Utilizar as chaves *public/private/n* especificadas no código C
      - https://www.di-mgt.com.au/rsa_alg.html

Interface entre Processador e Periféricos

- Trabalho 4 – parte 1
  - *CryptoMessage*
    - Parâmetros

Tempo ocioso após o envio de uma mensagem completa

Nome do arquivo texto contendo as mensagens. Cada linha corresponde a uma mensagem.

Chave usada para a encriptação

```
entity CryptoMessage is
    generic (
        MSG_INTERVAL      : integer;    -- Clock cycles
        FILE_NAME         : string := "UNUSED";
        PUBLIC_KEY        : type (integer/std_logic_vector);
        N                 : type (integer/std_logic_vector);
    );
    port (
        clk           : in std_logic;
        rst           : in std_logic;
        ack           : in std_logic;
        ...
```

Tomar como base o *CryptoMessage.vhd* disponível no *moodle*

Interface entre Processador e Periféricos

- Trabalho 4 – parte 1
  - Ligar o *CryptoMessage* ao *MIPS_uC*
    - *CryptoMessage* deve operar na metade da frequência do MIPS
  - Deve-se configurar adequadamente os bits da porta de E/S a fim que o *MIPS_uC* tenha acesso à interface do *CryptoMessage*
  - A saída *data_av* do *CryptoMessage* deve interromper o MIPS quando disponibiliza o primeiro byte da mensagem a ser transmitida (hardware)
  - Durante o tratamento da interrução, o MIPS deve fazer *polling* em *data_av* a fim de detectar um novo byte da mensagem (software), além de fazer *polling* em *eom* a fim de detectar o final da mensagem

Interface entre Processador e Periféricos

- Trabalho 4 – parte 1
  - Adicionar ao processador dois registradores especiais de 32 bits: *hi* e *lo*. Estes registradores devem armazenar o resultado das instruções de multiplicação e divisão
  - Implementar as instruções MULTU, DIVU, MFHI e MFLO
  - Para implementar as instruções MULTU e DIVU em VHDL no *datapath*, utilizar os operadores *, / e mod

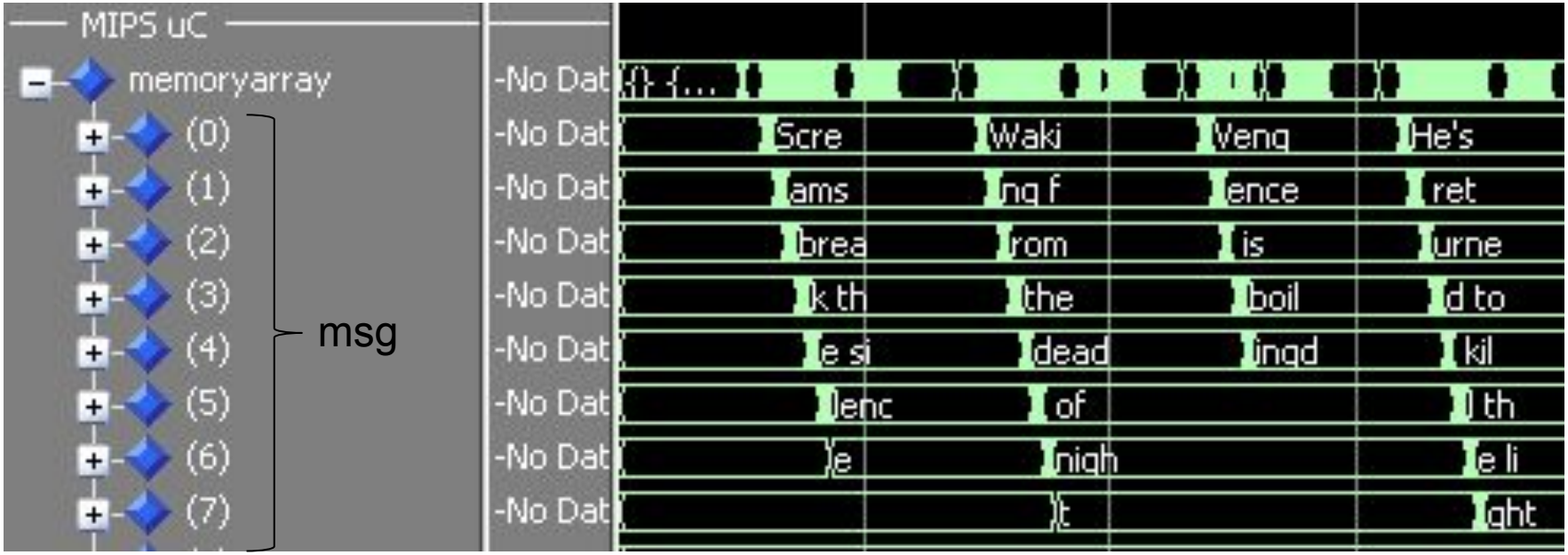
Interface entre Processador e Periféricos

- Trabalho 4 – parte 1
  - Aplicação
    - A aplicação principal executada pelo processador será o *bubbleSort*
      - Aumentar o tamanho do *array* para 50 elementos
    - Ao ser interrompido pelo *CryptoMessage*, o processador deve ler uma mensagem completa (*eom = 1*) e armazenar em um *array*
      - msg: .space 80 // Aloca 80 bytes
      - Este *array* deve ser sobreescrito quado o *CryptoMessage* enviar uma nova mensagem
    - O arquivo contendo as as mensagens enviadas pelo *CryptoMessage* será fornecidos junto com o *CryptoMessage* .vhd

Interface entre Processador e Periféricos

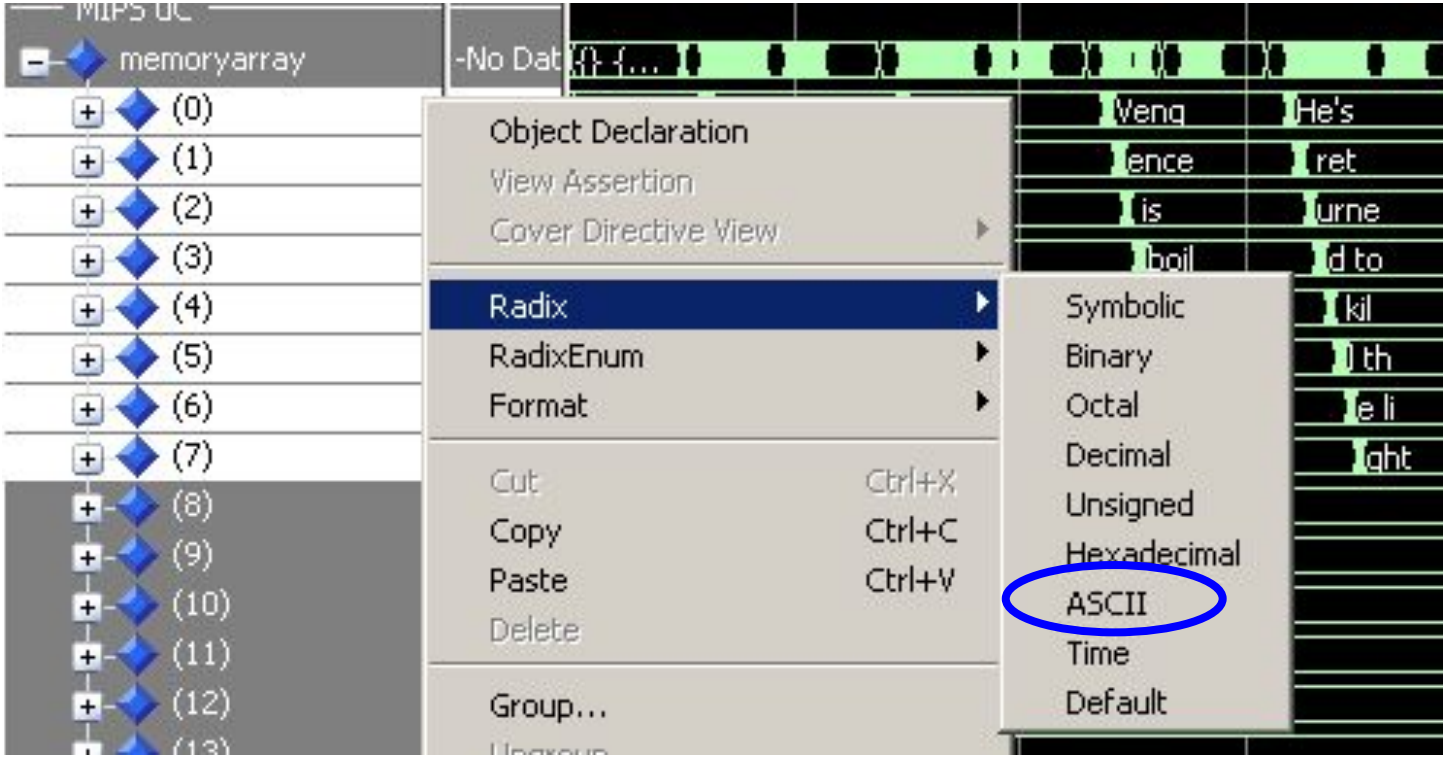
- Trabalho 4 – parte 1
  - Aplicação
    - IMPORTANTE: o armazenamento dos caracteres em memória deve ser feito de maneira que permita a leitura

Selecionar o formato ASCII na area de memória onde as mensagens são armazenadas
ModelSim: Radix→ASCII



Interface entre Processador e Periféricos

- Trabalho 4 – parte 1
  - Aplicação



Interface entre Processador e Periféricos

- Trabalho 4 – parte 1
  - Estrutura do código na memória de instruções