Arquitetura MIPS

Organização de Computadores Digitais – Projeto Final

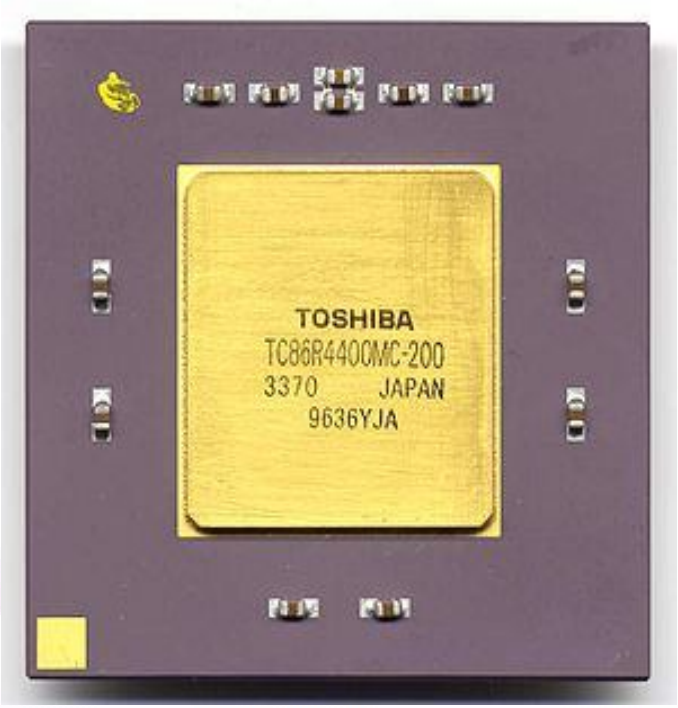
Melina Brilhadori nºUSP 6412633
Murilo Galvão Honorio nºUSP 6411927
4º Semestre - Turma 02 - Matutino

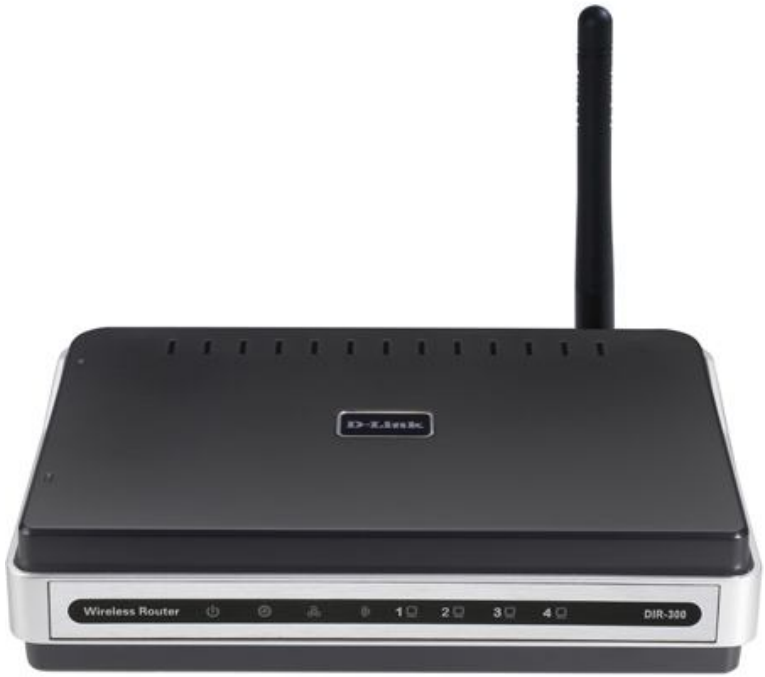
Tópicos

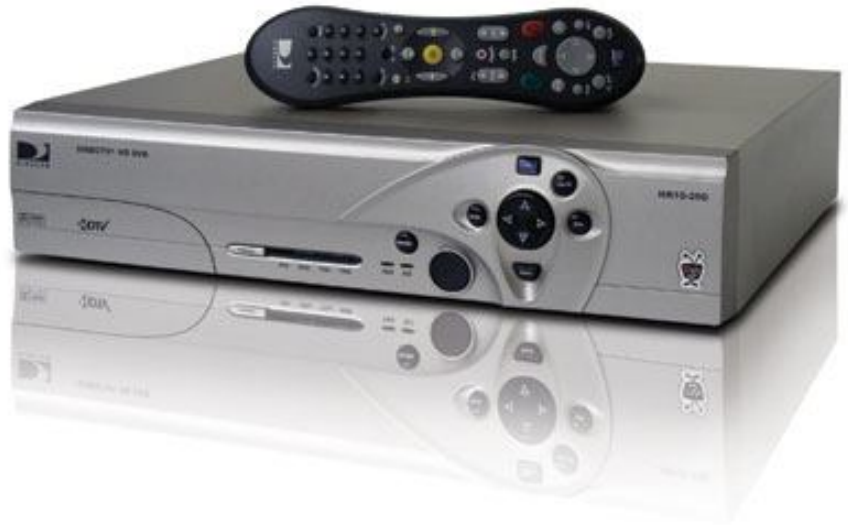
- Onde o MIPS é utilizado?
- SPIM x MIPS
- Resolução de exercício

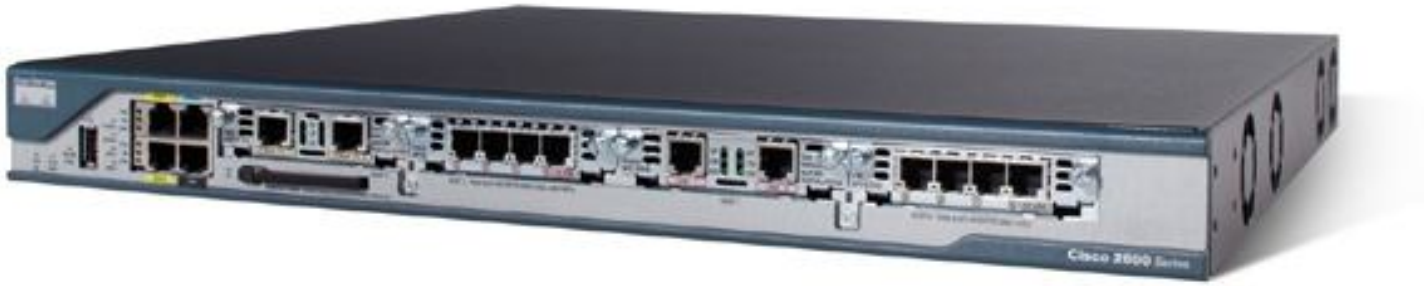
Onde o MIPS é utilizado?

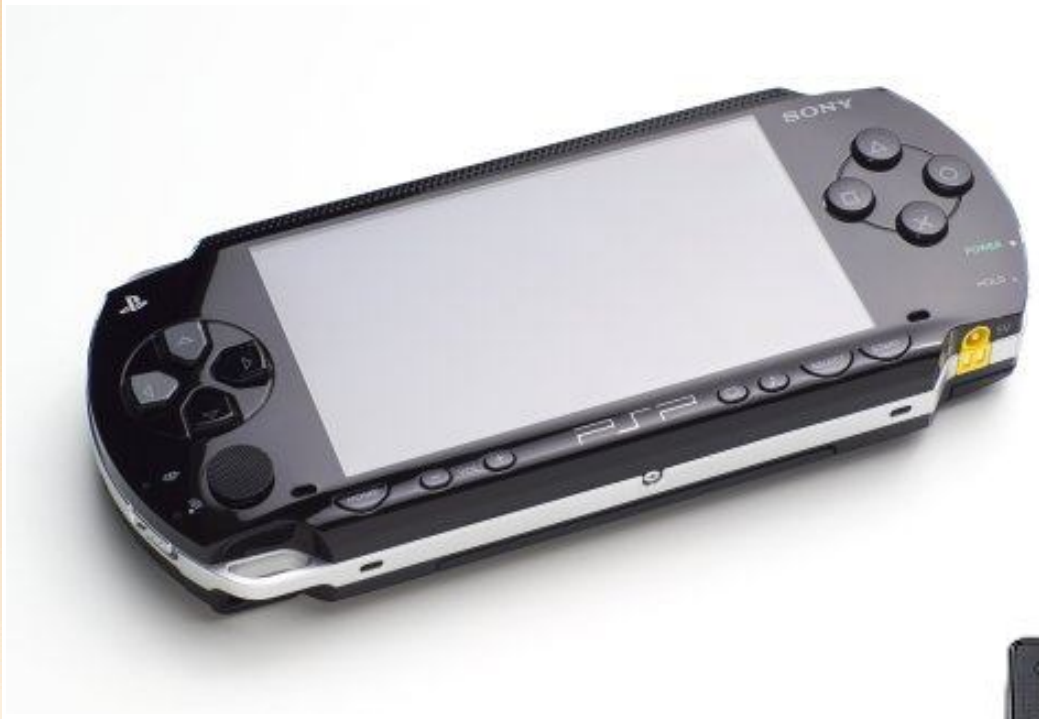
- Gravadores de Vídeo Digital (DVR)
- Roteadores Cisco
- Workstations SGI
- Roteadores domésticos
- Videogames Playstation I, II, Portable, Nintendo 64
- Netbook Alpha 400
- Milhares de sistemas dedicados (embedded)

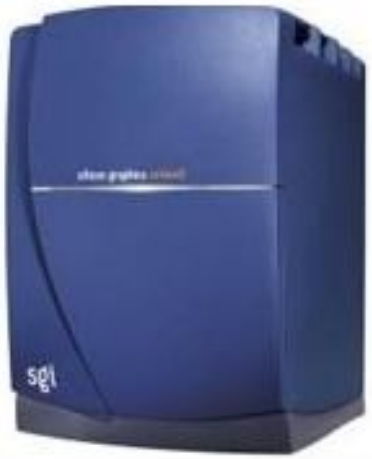


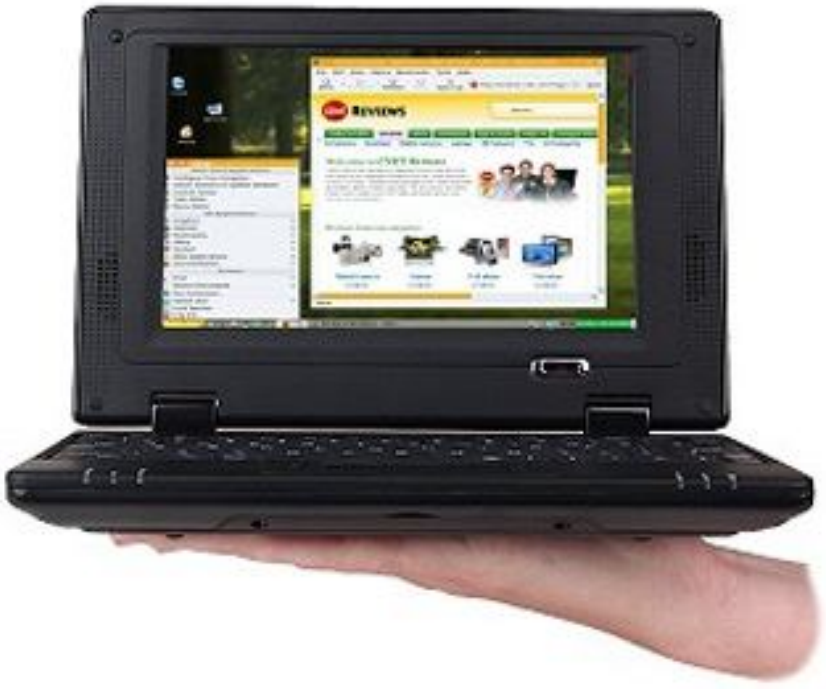

D-Link
Wireless Router
1
2
3
4
DIR-300

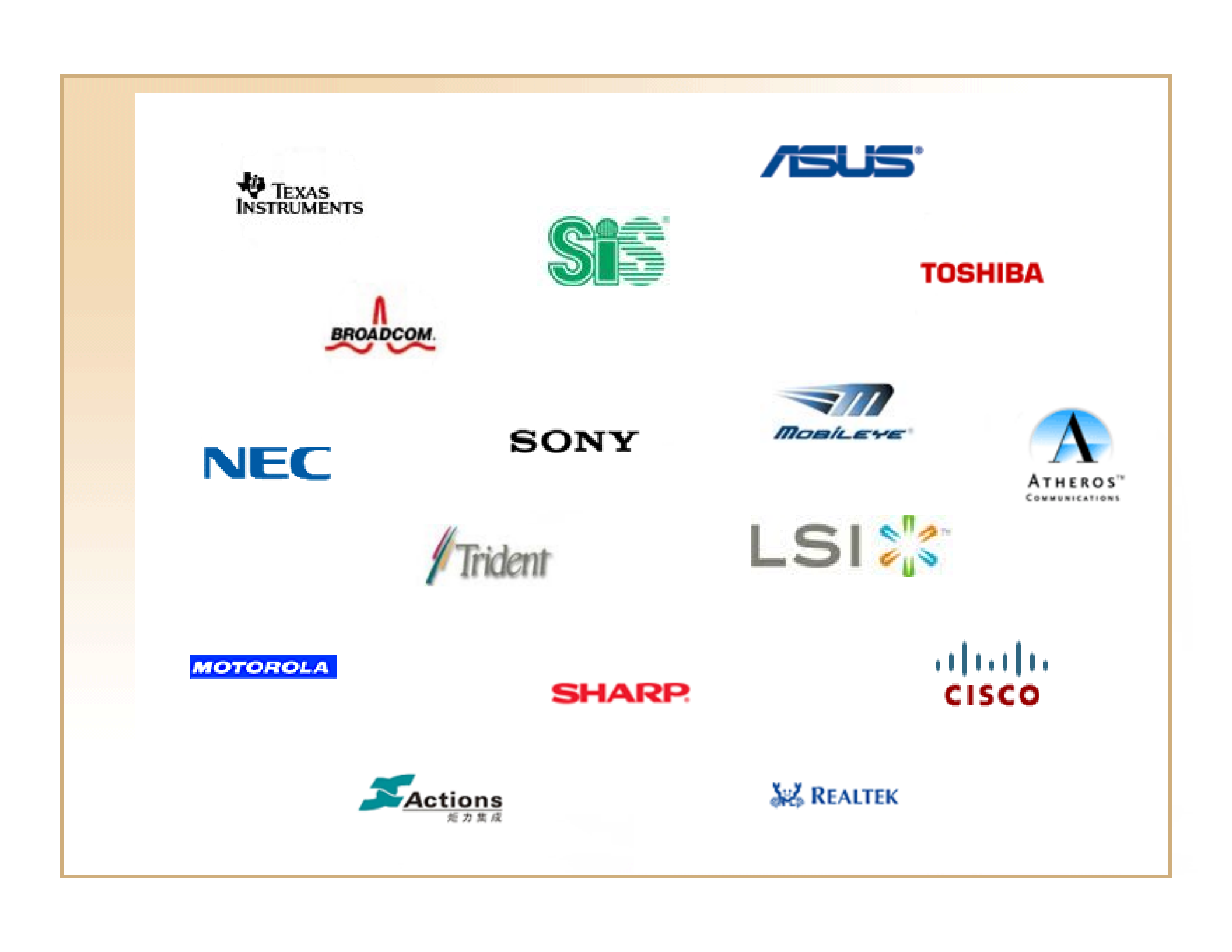



Cisco 2600 Series


SONY


sgi




Texas Instruments
ASUS
SiS
TOSHIBA
BROADCOM.
MOBILEYE
NEC
SONY
ATHEROS
COMMUNICATIONS
Trident
LSI
MOTOROLA
SHARP
CISCO
Actions
REALTEK

SPIM x MIPS

- Tempos das instruções: diferentes do real

- Exceções: as instruções de ponto flutuante não detectam muitas condições de erro, o que deveria causar exceções em uma máquina real.

SPIM x MIPS

- Memória: não existem caches nem atrasos no acesso

- Operações: não reflete com precisão os atrasos na operação de ponto flutuante ou nas instruções de multiplicação e divisão.

Exercício proposto:

- Escreva e teste um programa que **leia um inteiro positivo** usando as chamadas do sistema do SPIM. **Se** o inteiro **não for positivo**, o programa deverá terminar com a **mensagem "Entrada inválida"**; **caso contrário**, o programa deverá **imprimir** os nomes dos **dígitos dos inteiros por extenso**, delimitados por exatamente um espaço.
- Por exemplo, se o usuário informou "728", a saída deverá ser "Sete Dois Oito".

Registradores

- $s0 – número inteiro digitado pelo usuário;
- $s1 - número total de dígitos;
- $a0 - parâmetro de syscall;
- $v0 - parâmetro de syscall;
- $a1 - argumento da sub-rotina *numtexto* (dígito);
- $a2 - endereço de n[0];
- $t0 – inteiro utilizado temporariamente nas divisões sucessivas;
- $t2 - armazena um dígito isolado;
- $t3 - endereço temporário da sub-rotina *load_num*, índice;
- $sp - topo da pilha;
- $ra - endereço de retorno.

Seção de dados (.data)

.asciiz – diretiva que cria strings:

- aut_msg:"Exercico A.8 de OCD,\npor Melina Brilhadori e Murilo Honorio.\n";
- ins_msg:"Digite um numero inteiro: ";
- inv_msg:"Entrada invalida";
- espaco: " ";
- n_zero:"Zero" até n_nome:"Nove“;

.word – diretiva que cria vetores (32 bits):

- n: .word 0,0,0,0,0,0,0,0,0,0

→ n[10], array de dez dígitos

Sub-rotina load_num

- Carrega os textos dos digitos por extenso no vetor n[]

```
load_num:
    la    $a2, n          # endereco de n[0]
    la    $t3, n_zero     # endereco de "Zero"
    sw    $t3, 0($a2)     # grava o endereco de "Zero" em n[0]
    la    $t3, n_ummm     # endereco de "Um"
    sw    $t3, 4($a2)     # grava o endereco de "Um" em n[1]
    ...
    la    $t3, n_nove     # endereco de "Nove"
    sw    $t3, 36($a2)    # grava o endereco de "Nove" em n[9]
    jr    $ra             # retorna
```

Label inval

- O programa redireciona para este label caso receba como entrada um inteiro negativo

```
inval:
    la      $a0, inv_msg      # coloca o endereço de inv_msg em $a0
    li      $v0, 4            # syscall = 4: "print_string"
    syscall                   # mensagem: "Entrada invalida"
    b       fim               # desvia para o fim
```

Sub-rotina separa

- Separa cada dígito do numero armazenado $s0, colocando na pilha

```
separa:
    li      $t0, 0              # inicializa $t0 = 0
    add     $t0, $t0, $s0       # $t0 = $s0 -
repete:
    rem     $t2, $t0, 10        # $t2 = ($t0 % 10), separa o digito
    addi    $sp, $sp, -4        # push: ajusta $sp
    sw      $t2, 0($sp)         # push: empilha digito
    addi    $s1, $s1, 1         # digitos++ (total de dígitos)
    div     $t0, $t0, 10        # $t0 = ($t0 / 10), exclui o digito
    bgtz    $t0, repete         # se ($t0 > 0) repete
    jr      $ra                 # retorna
```

Sub-rotina numtexto

- Imprime na tela o dígito lido do registrador $a1 (que foi retirado da pilha no main)

```
numtexto:
    la      $a2, n          # n[0]
    mul     $t3, $a1, 4     # i = digito*4 (posicionar)
    addu    $a2, $a2, $t3   # array[i]
    lw      $a0, 0($a2)     # $a0 = array[i]
    li      $v0, 4          # syscall = 4: "print_string"
    syscall                 # imprime o numero
    jr      $ra             # retorna
```

- Nas multiplicações utiliza-se 4 porque os saltos são em bytes (4 bytes = 32 bits, tamanho da word)

Main

```
main:
    la      $a0, aut_msg     # carrega endereço da mensagem
    li              $v0, 4           # syscall = 4: "print_string"
    syscall                          # mensagem: Exercicio/autores
    la      $a0, ins_msg      # carrega endereço da mensagem
    li              $v0, 4           # syscall = 4: "print_string"
    syscall                          # mensagem: Digite numero

    li              $v0, 5           # syscall = 5: "read_int"
    syscall                          # lê inteiro
    move            $s0, $v0         # move o inteiro lido para $s0
    bltz            $s0, inval       # se ($s0 < 0), vai para inval
    jal             separa           # desvia para 'separa'
    jal             load_num         # desvia para 'load_num'
```

Main (cont.), labels imprime, fim

```
imprime:
    lw          $a1, ($sp)      # pop: desempilha dígito para
                                # argumento $a1, utilizado por
                                # numtexto
    addi        $sp, $sp, 4     # pop: ajusta $sp
    jal         numtexto        # desvia para 'numtexto'
    addi        $s1, $s1, -1    # dígitos-- (decrementa total)
    beq         $s1, 0, fim     # se ($s1 == 0) fim
    la          $a0, espaco
    li          $v0, 4          # syscall = 4: "print_string"
    syscall                     # imprime na tela um espaço
    b           imprime         # repete imprime

fim:    li      $v0, 10         # syscall = 10: "exit"
        syscall                 # sair
```