AC2 - Exercício 09 - João Comini César de Andrade

Atividade 1

O que acontece? O problema ilustra a importância do caractere nulo (\0 ou 0x00) para marcar o fim de uma string.
Primeira impressão de str2: O código preenche str2 com 20 letras 'A' (ASCII 65). Como str2 foi declarado com .space 20, não há garantia de que o byte seguinte seja zero. Imediatamente após str2 na memória está str3 ("Hello World"). Como não há um \0 após os 20 'A's, a ecall 4 continua imprimindo o que estiver na memória até achar um zero.
Resultado: Imprime os 20 'A's seguidos imediatamente por "Hello World".
Segunda impressão de str2: A instrução sb zero, 19(a1) coloca um terminador nulo na última posição do espaço alocado.
Resultado: Agora a ecall 4 para onde deve, imprimindo apenas 19 'A's (o 20º foi substituído por zero).

Atividade 2

```
.data
str_origem: .string "String original\n"
str_destino: .space 50 # Espaço suficiente

.text
.globl main
main:
    la a0, str_destino
    la a1, str_origem
    call strcpy

    # Imprime destino para provar a cópia
    li a0, 4
    la a1, str_destino
    ecall

    # Encerra
    li a0, 10
    ecall
```

```
# char * strcpy(char *s1 (a0), char *s2 (a1))
strcpy:
    mv t0, a0         # Salva inicio de s1 para retornar se necessario
loop_cpy:
    lbu t1, 0(a1)     # Lê byte de s2
    sb t1, 0(a0)      # Grava byte em s1

    beqz t1, fim_cpy # Se copiou o 0, terminou

    addi a0, a0, 1    # Avança ponteiro s1
    addi a1, a1, 1    # Avança ponteiro s2
    j loop_cpy

fim_cpy:
    mv a0, t0         # Restaura a0 original (convenção C)
    ret
```

Atividade 3

```
# char * gets(char *s (a0))
gets:
    mv t0, a0         # t0 guarda o ponteiro atual onde escreveremos

    # 1. Habilitar teclado
    li a0, 0x130
    ecall

loop_gets:
    # 2. Verificar se tem char (Polling)
    li a0, 0x131
```

```
    ecall

    # Analisa retorno em a0
    # 0 = Acabou tudo (EOF)
    # 1 = Nada ainda, tente de novo
    # 2 = Tem char em a1

    li t1, 1
    beq a0, t1, loop_gets  # Se 1, continua esperando
    beqz a0, fim_gets      # Se 0, sai (opcional, segurança)

    # Se chegou aqui, a0 é 2. O char está em a1.

    # Verifica 'Enter' (Newline \n = 10)
    li t2, 10
    beq a1, t2, fim_gets

    # Armazena o char
    sb a1, 0(t0)
    addi t0, t0, 1   # Avança ponteiro
    j loop_gets

fim_gets:
    sb zero, 0(t0)   # Adiciona \0 no final
    # Retorna o endereço original (já estava em a0 no inicio,
    # mas a chamada de sistema alterou a0. Precisamos recuperar o
inicio se salvamos antes
    # ou assumir void. Para seguir C, idealmente restauramos o inicio).
    ret
```

Atividade 4

```
# char * fgets(char *s (a0), int N (a1))
```

```
fgets:
    mv t0, a0         # t0 = ponteiro escrita
    mv t3, a1         # t3 = Limite N
    addi t3, t3, -1   # Reserva espaço para o \0

    # Habilitar teclado
    li a0, 0x130
    ecall

loop_fgets:
    beqz t3, fim_fgets # Se contador chegou a 0, para de ler

    li a0, 0x131
    ecall

    li t1, 1
    beq a0, t1, loop_fgets
    beqz a0, fim_fgets

    # Char em a1
    li t2, 10
    beq a1, t2, fim_fgets

    sb a1, 0(t0)
    addi t0, t0, 1    # Avança ponteiro
    addi t3, t3, -1   # Decrementa espaço restante
    j loop_fgets

fim_fgets:
    sb zero, 0(t0)    # Adiciona \0
    ret
```

Atividade 5

```
# int strlen(char *s (a0)) -> Retorna tamanho em a0
strlen:
    mv t0, a0        # Salva inicio
loop_len:
    lbu t1, 0(a0)
    beqz t1, fim_len
    addi a0, a0, 1
    j loop_len
fim_len:
    sub a0, a0, t0   # Endereço final - Endereço inicial = Tamanho
    ret

# int strcmp(char *s1 (a0), char *s2 (a1)) -> <0, 0, >0
strcmp:
loop_cmp:
    lbu t0, 0(a0)
    lbu t1, 0(a1)
    sub t2, t0, t1   # t2 = s1[i] - s2[i]
    bnez t2, fim_cmp # Se diferente, retorna a diferença

    beqz t0, fim_cmp # Se s1[i] é 0 (e são iguais), acabou as strings

    addi a0, a0, 1
    addi a1, a1, 1
    j loop_cmp
fim_cmp:
    mv a0, t2
    ret

# char * strcat(char *s1 (a0), char *s2 (a1))
strcat:
    mv t3, a0        # Salva inicio de s1 para retorno
```

```
    # 1. Achar o fim de s1
loop_find_end:
    lbu t0, 0(a0)
    beqz t0, start_cat
    addi a0, a0, 1
    j loop_find_end

    # 2. Copiar s2 para o final de s1
start_cat:
    lbu t0, 0(a1)
    sb t0, 0(a0)
    beqz t0, fim_cat
    addi a0, a0, 1
    addi a1, a1, 1
    j start_cat

fim_cat:
    mv a0, t3        # Restaura inicio de s1
    ret
```

Atividade 6

```
.data
buffer: .space 100
msg:    .string "Digite algo para inverter: "
newline:.string "\n"

.text
.globl main
main:
    # 1. Imprime mensagem
    li a0, 4
```

```
    la a1, msg
    ecall

    # 2. Lê a string do teclado (usando nosso fgets)
    la a0, buffer    # Onde guardar
    li a1, 100       # Tamanho máximo
    call fgets

    # 3. Inverte a string (usando nosso strrev)
    la a0, buffer
    call strrev

    # 4. Imprime o resultado (já invertido no buffer)
    li a0, 4
    la a1, buffer
    ecall

    # 5. Nova linha e Sair
    li a0, 4
    la a1, newline
    ecall

    li a0, 10
    ecall

# ================================================================
# FUNÇÃO STRREV: Inverte uma string no próprio lugar (in-place)
# void strrev(char *s (a0))
# ================================================================
strrev:
    # Salva RA e S0 na pilha (S0 para segurar o ponteiro inicial)
    addi sp, sp, -8
    sw   ra, 4(sp)
```

```
    sw   s0, 0(sp)

    mv   s0, a0          # s0 = ponteiro INICIO (preservado)

    # Chama strlen para saber onde é o fim
    # a0 já está com o endereço da string
    call strlen          # Retorna tamanho em a0

    # Se tamanho for 0 ou 1, não precisa inverter
    li   t0, 1
    ble  a0, t0, fim_rev

    # Calcula ponteiro FIM
    # fim = inicio + tamanho - 1 (para pular o \0 e pegar o último
char)
    add  t1, s0, a0      # t1 = inicio + tamanho (aponta para o \0)
    addi t1, t1, -1      # t1 = aponta para o último caractere útil

    mv   t0, s0          # t0 = ponteiro INICIO (cópia para o loop)

loop_rev:
    bge  t0, t1, fim_rev # Se inicio >= fim, paramos

    # Troca (Swap)
    lbu  t2, 0(t0)       # carrega char do inicio
    lbu  t3, 0(t1)       # carrega char do fim

    sb   t3, 0(t0)       # grava fim no inicio
    sb   t2, 0(t1)       # grava inicio no fim

    addi t0, t0, 1       # inicio++
    addi t1, t1, -1      # fim--
    j    loop_rev
```

```
fim_rev:
    # Restaura pilha
    lw   s0, 0(sp)
    lw   ra, 4(sp)
    addi sp, sp, 8
    ret

# ================================================================
# FUNÇÃO STRLEN: Retorna o tamanho da string
# int strlen(char *s (a0))
# ================================================================
strlen:
    mv   t0, a0          # Salva inicio para calculo final
loop_len:
    lbu  t1, 0(a0)       # Lê byte
    beqz t1, calc_len    # Se 0, fim
    addi a0, a0, 1       # Próximo
    j    loop_len
calc_len:
    sub  a0, a0, t0      # Tamanho = End. Atual - End. Inicial
    ret

# ================================================================
# FUNÇÃO FGETS: Lê string do teclado com limite (Polling no Venus)
# char * fgets(char *s (a0), int N (a1))
# ================================================================
fgets:
    mv   t0, a0          # t0 = ponteiro de escrita
    mv   t3, a1          # t3 = Limite N
    addi t3, t3, -1      # Deixa espaço para o \0

    # Habilitar teclado (ecall 0x130)
```

```
    li   a0, 0x130
    ecall

loop_fgets:
    beqz t3, fim_fgets   # Se encheu o buffer, para

    # Verifica se tem char (Polling - ecall 0x131)
    li   a0, 0x131
    ecall

    li   t1, 1
    beq  a0, t1, loop_fgets # Se 1, nada digitado ainda, repete
    beqz a0, fim_fgets      # Se 0, entrada fechada

    # Se a0 == 2, tem char em a1
    li   t2, 10             # ASCII para \n (Enter)
    beq  a1, t2, remove_nl  # Se for Enter, para a leitura

    sb   a1, 0(t0)          # Salva o char no buffer
    addi t0, t0, 1          # Avança ponteiro
    addi t3, t3, -1         # Decrementa limite
    j    loop_fgets

remove_nl:
    # Opcional: Se quiser pular o \n e não gravar na string, vem direto
pra cá.
    # Se quiser gravar o \n, faça o sb antes. Aqui optamos por não
gravar o \n.

fim_fgets:
    sb   zero, 0(t0)        # Adiciona o \0 final
    ret
```