PCS3432 - Laboratório de Processadores

Tarefa - E8

Bruno Mariz - 11261826

Códigos utilizado na tarefa:

11261826.c:

```c
#include <stdio.h>

extern int int2str(int a, char *string);
void impstr(char *pont);

int main()
{
    __asm__("pronto: nop\n\t");

    char strNUSP[100];

    int NUSP = 826;
    int2str(NUSP, strNUSP);
    impstr(strNUSP);
    return 0;
}

void impstr(char *pont)
{
    __asm__("inic_imprime: nop\n\t");
    if (pont[0] != 0)
    {
        // imprime caracter *pont
        printf("%c\n", pont[0]);
        impstr(pont + 1);
    }
    __asm__("fim_imprime: nop\n\t");
    return;
}
```

int2str.s:

```
    .text
    .globl  int2str

teste:
```

```
    @ Inicializar parametros
    LDR r0, =826 @ dividendo
    @ LDR r0, =832 @ dividendo
    LDR r1, =pontstr @ string

    BL int2str
    B fim

int2str:
    @ Transforma inteiro em string

    @ Salva contexto
    STMFD sp!, {fp, ip, lr}

    @ Inicializa contador
    MOV r7, #0

    @ While r0 != 0
    while_int2str:
    CMP      r0, #0
    BNE      while_int2str_not_zero
    @ Adiciona 0 ao fim da string
    RSB r6, r7, #2
    STRB r5, [r1, r6]
    LDMFD  sp!, {fp, ip, lr}
    MOV    pc, lr

    while_int2str_not_zero:
    @ Divide o parametro por 10 e pega proximo numero
    STMFD sp!, {r1, r7}
    MOV r1, #10
    BL divisao
    LDMFD sp!, {r1, r7}

    @ Guarda proximo digito como char
    ADD r5, r5, #0x30
    RSB r6, r7, #2
    STRB r5, [r1, r6]

    @ Atualiza dividendo com o quociente anterior
    MOV r0, r3

    @ Incrementa contador
    ADD r7, r7, #1

    BAL      while_int2str




divisao:
    STMFD sp!, {fp, ip, lr}
    MOV r3, #0x0 @ quociente
```

```
    @ Inicializar resto com dividendo
    MOV r5, r0 @ resto
    @ Salva divisor original
    MOV r7, r1

    @ Mover bits do divisor para a esquerda
    BL mover_bits_divisor

    @ Comparar dividendo divisor
    if_compara_dividendo_divisor:
    CMP r5, r1
    BGE cabe
    nao_cabe:
    @ Concatena 0 ao fim do quociente
    MOV r3, r3, LSL #1
    B desloca_divisor
    cabe:
    @ Subtrai divisor do dividendo
    SUB r5, r5, r1
    @ Concatena 1 ao fim do quociente
    MOV r3, r3, LSL #1
    ADD r3, r3, #1
    desloca_divisor:
    MOV r1, r1, LSR #1
    until_compara_divisor_com_original:
    CMP r1, r7
    BGE if_compara_dividendo_divisor

    @ fim da funcao
    LDMFD sp!, {fp, ip, lr}
    MOV pc, lr

mover_bits_divisor:
    @ Checa se primeiro bit do r1 eh 1
    TST     r1, #0x40000000
    BEQ    a_mbd @ se for 0, faz o salto
    MOV     pc, lr @ se for 1, volta para o programa
a_mbd:
    MOV r1, r1, LSL #1  @ desloca bits do r1 para a esquerda
    B mover_bits_divisor

@ Fim do programa
fim:
MOV r0, #0x18
LDR r0, =0x20026
SWI 0x0


.data
    pontstr: .ascii ""
```

Os códigos foram compilados utilizando o comando:

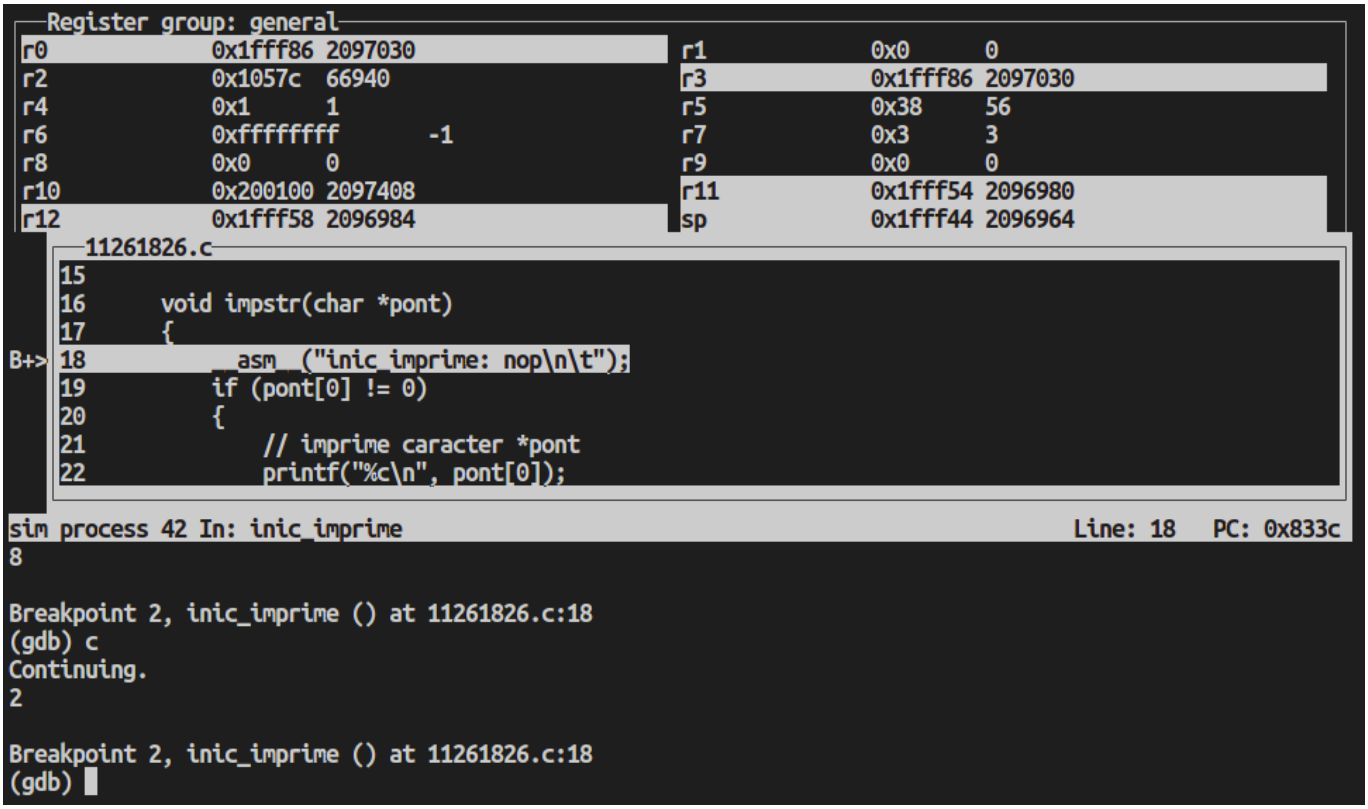
```
gcc int2str.s 11261826.c
```

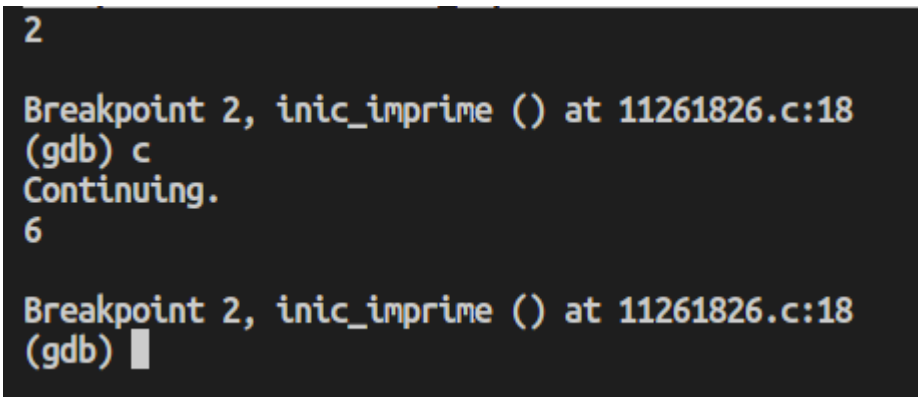
Ao notar que utilizando a função `printf("%c", pont[0]);` (sem `\n`) os digitos eram impressos todos de uma vez, foi utilizada a função `printf("%c\n", pont[0]);` para que fosse impresso um digito por vez.

Ao executar o programa no gdb com o comando

```
gdb a.out
```

foram observados os digitos sendo impressos um a um:





Foram então executados os comandos do enunciado em 8.3.4 e foi gerado o arquivo de saída a seguir:

11261826.txt:

```
warning: Current output protocol does not support redirection
Continuing.
0x1fff6c:   0x001fff84  0x001ffff4  0x001fff80  0x0000831c
0x1fff7c:   0x00008338  0x3800033a  0x00363238  0x00000000
0x1fff8c:   0x00000000  0x00000000  0x00000000  0x00000000
0x1fff9c:   0x00000000  0x00000000  0x00000000  0x00000000
0x1fffac:   0x00000000  0x00000000  0x00000000  0x00000001
Continuing.
8
0x1fff58:   0x001fff85  0x001fff7c  0x001fff6c  0x00008378
0x1fff68:   0x00008338  0x001fff84  0x001ffff4  0x001fff80
0x1fff78:   0x0000831c  0x00008338  0x3800033a  0x00363238
0x1fff88:   0x00000000  0x00000000  0x00000000  0x00000000
0x1fff98:   0x00000000  0x00000000  0x00000000  0x00000000
Continuing.
2
0x1fff44:   0x001fff86  0x001fff68  0x001fff58  0x00008378
0x1fff54:   0x00008338  0x001fff85  0x001fff7c  0x001fff6c
0x1fff64:   0x00008378  0x00008338  0x001fff84  0x001ffff4
0x1fff74:   0x001fff80  0x0000831c  0x00008338  0x3800033a
0x1fff84:   0x00363238  0x00000000  0x00000000  0x00000000
Continuing.
6
0x1fff30:   0x001fff87  0x001fff54  0x001fff44  0x00008378
0x1fff40:   0x00008338  0x001fff86  0x001fff68  0x001fff58
0x1fff50:   0x00008378  0x00008338  0x001fff85  0x001fff7c
0x1fff60:   0x001fff6c  0x00008378  0x00008338  0x001fff84
0x1fff70:   0x001ffff4  0x001fff80  0x0000831c  0x00008338
Continuing.
0x1fff30:   0x001fff87  0x001fff54  0x001fff44  0x00008378
0x1fff40:   0x00008338  0x001fff86  0x001fff68  0x001fff58
0x1fff50:   0x00008378  0x00008338  0x001fff85  0x001fff7c
0x1fff60:   0x001fff6c  0x00008378  0x00008338  0x001fff84
0x1fff70:   0x001ffff4  0x001fff80  0x0000831c  0x00008338
Continuing.
0x1fff44:   0x001fff86  0x001fff68  0x001fff58  0x00008378
0x1fff54:   0x00008338  0x001fff85  0x001fff7c  0x001fff6c
0x1fff64:   0x00008378  0x00008338  0x001fff84  0x001ffff4
0x1fff74:   0x001fff80  0x0000831c  0x00008338  0x3800033a
0x1fff84:   0x00363238  0x00000000  0x00000000  0x00000000
Continuing.
0x1fff58:   0x001fff85  0x001fff7c  0x001fff6c  0x00008378
0x1fff68:   0x00008338  0x001fff84  0x001ffff4  0x001fff80
0x1fff78:   0x0000831c  0x00008338  0x3800033a  0x00363238
0x1fff88:   0x00000000  0x00000000  0x00000000  0x00000000
0x1fff98:   0x00000000  0x00000000  0x00000000  0x00000000
Continuing.
0x1fff6c:   0x001fff84  0x001ffff4  0x001fff80  0x0000831c
0x1fff7c:   0x00008338  0x3800033a  0x00363238  0x00000000
0x1fff8c:   0x00000000  0x00000000  0x00000000  0x00000000
0x1fff9c:   0x00000000  0x00000000  0x00000000  0x00000000
0x1fffac:   0x00000000  0x00000000  0x00000000  0x00000001
Breakpoint 4 at 0x82c8: file int2str.s, line 102.
```

```
Continuing.
```

A execução do programa foi mostrada ao professor, que aprovou o resultado.