CURSO:	ENGENHARIA DE SOFTWARE	
DISCIPLINA:	Fundamentos de Arquitetura de Computadores	TURMA: A
SEMESTRE:	1°/2019	
PROFESSOR:	Tiago Alves da Fonseca	
ALUNOS:	Sara Conceição de Sousa Araújo Silva 16/0144752 Shayane Marques Alcântara 16/0144949	

Relatório Projeto
Aritmética de inteiros

Introdução

Assembly é a linguagem de montagem que faz uma abstração da linguagem de máquina usada em dispositivos computacionais. Cada declaração em Assembly produz uma instrução de máquina, fazendo uma correspondência um-para-um. A programação em linguagem de montagem é preferível em relação a linguagem de máquina porque a utilização de nomes simbólicos e endereços simbólicos em vez de binários ou hexadecimais facilita a memorização e o aprendizado das instruções. Então, o programador de Assembly só precisa se lembrar dos nomes simbólicos porque o compilador assembler os traduz para instruções de máquina.

Neste trabalho, duas estudantes de Fundamentos de Arquitetura de Software praticaram instruções em Assembly para implementar a fatoração de números inteiros e em seguida imprimir os fatores correspondentes.

Objetivos

1) Exercitar conceitos da linguagem de montagem para arquitetura MIPS, especialmente aqueles referentes à implementação de solução de problemas em aritmética inteira.
2) Interagir com ferramentas de desenvolvimento para criação, gerenciamento, depuração e testes de projeto de aplicações.

Referências Teóricas

David A. Patterson, John L. Hennessy , Computer Organization and Design: the Hardware/Software Interface, The Morgan Kaufmann series in computer architecture and design ,5th ed

Material Necessário

- Computador com sistema operacional programável
    - Manjaro Linux;
    - Ubuntu 18.10.
- Ambiente de simulação para arquitetura MIPS: MARS.

Roteiro

1. Revisão de técnicas e ferramentas de desenvolvimento usando linguagem de montagem MIPS.Colete o material acompanhante do roteiro do trabalho a partir do Moodle da disciplina e estude os princípios e técnicas de desenvolvimento de aplicações usando linguagem de montagem MIPS
2. Realizar as implementações solicitadas no questionário do trabalho.

Procedimentos

1. Ler, a partir do terminal, um número inteiro positivo maior que 1 e menor que 32768, em seguida,
2. Realizar a fatoração do número
3. Imprimir os fatores que foram usados

Exemplos de entrada e saídas:

```
9
saída:
3
3
____
-1
saída:
Entrada invalida.
____
0
saída:
Entrada invalida.
____
32768
saída:
Entrada invalida.
____
6
Saída:
```

```
2
3
____
37
saída:
37
8017
saída
8017
____
8016
saída:
2
2
2
2
3
167
```

Solução

Primeiro, o programa escrito no simulador MARS cria a mensagem que será exibida quando houver uma entrada inválida e uma quebra de linha.

```
        .data

invalida: .asciiz "Entrada invalida.\n"          # Mensagem que será exibida caso a entrada não esteja entre 1 e 32768
quebra_linha: .asciiz "\n"                       # Quebra de linha
```

Em seguida, começa a rotina principal *main*. Nela é feita a requisição do número inteiro de entrada a partir do terminal. Ocorre a validação do número de entrada, em que

este deve estar no intervalo de maior que 1 e menor 32768. Caso a entrada esteja fora desse intervalo, o programa desvia para a *se_invalida,* que faz o devido tratamento. Também é colocado o número de entrada ($s0)num registrador auxilar ($t7).

```
main:
        li $v0, 5                       # Leitura de um inteiro
        syscall                         # Faz a chamada do sistema

        move $s0, $v0                   # Coloca o conteúdo de $v0 em $s0

        slti $t1, $s0, 2                # Verifica se $s0 é menor que 2
        bne $t1, $zero, se_invalida     # Se for menor que 2, desvia para rotina de tratamento de entrada invalida

        bge $s0, 32768, se_invalida     # Verifica se $s0 é maior ou igual a 32768, se for, desvia para rotina que trata entra

        move $t7, $s0                   # $t7 vai ser usado como registrador auxiliar
```

Em seguida, começa a rotina que verifica se o único primo par, o 2, é um fator do número. O auxiliar é dividido por 2, se o resto for 0, então 2 é um fator e precisa ser impresso. Se não, o número 3 é usado como primeiro primo ímpar possível($t4) e o registrador $t5 guarda o número 9 ($3^2$), pois se um número é fator, o quadrado dele deve ser igual ou menor ao número que está sendo fatorado, então pula para rotina que verifica fatores ímpares.

```
fatores_2:                              # Rotina que verifica se 2 é fator do número, o único primo par
        div $t7, $t7, 2                 # Divide o número por 2
        mfhi $t3                        # Salva o resto da divisão no registrador $t3
        beq $t3, $zero, imprime_2       # Se o resto for igual 0 então 2 é fator e precisa ser impresso

        add $t4, $t4, 3                 # primeiro primo ímpar possível
        add $t5, $t5, 9                 # t5 = 3²
        j fatores_impares               # Se 2 não for mais um fator, desvia para rotina de possíveis fatores ímpares
```

Rotina para impressão do fator 2. Nela também é atualizado o registrador $s0 com o número já dividido.

```
imprime_2:                              # Rotina para imprimir o número 2 como fator
        li $v0, 1                       # Impressão de um inteiro
        la $a0, 2                       # Imprime 2
        syscall                         # Faz a chamada do sistema

        li $v0, 4                       # Impressão de string
        la $a0, quebra_linha            # Imprime quebra linha a cada 2 impresso
        syscall                         # Faz a chamada do sistema

        move $s0, $t7                   # Atualiza o valor de $s0

        j fatores_2                     # Retorna para fator_2 para verificar se há mais fatores 2 para o número
```

Rotina que verifica se o primo atual ($t4) é divisor do número, se for, ele precisa ser impresso, se não, pula para rotina que calcula o próximo primo.

```
fatores_impares:                        # Verifica os possíveis fatores ímpares
        move $t7, $s0                   # Atualiza o $t7
        div $t7, $t7, $t4               # Divide o número por pelo primo atual
        mfhi $t3                        # Salva o resto da divisão no registrador $t3
        beq $t3, $zero, imprime_fator_impar     # Se o resto for igual 0 então $t4 é fator e precisa ser impresso

        j proximo_primo                 # Se não, calcula o próximo primo
```

Rotina para impressão dos fatores ímpares. Nela também é atualizado o registrador $s0, e se o quadrado do fator ($t5) for menor ou igual a ele, volta para fatores_impares.

```
imprime_fator_impar:                    # Rotina para imprimir o fator ímpar
        li $v0, 1                       # Impressão de um inteiro
        la $a0, ($t4)                   # Imprime o primo atual
        syscall                         # Faz a chamada do sistema

        li $v0, 4                       # Impressão de string
        la $a0, quebra_linha            # Imprime quebra de linha
        syscall                         # Faz a chamada do sistema

        div $s0, $s0, $t4               # Divide $s0 pelo primo atual

        ble $t5, $s0, fatores_impares   # Se $s0 ainda for menor que o primo ao quadrado, volta para fazer a verificação
```

Função que calcula o próximo primo. Como os números primos restantes são ímpares, soma 2 ao primo atual ($t4) e atualiza o quadrado no $t5, depois é verificado se o quadrado do primo é menor ou igual ao $s0 para desviar para *fatores_impares*.

```
proximo_primo:                          # Rotina que calcula o próximo primo
        addi $t4, $t4, 2                # Como os próximos primos sempre serão ímpares, $t4 pula de 2 em 2
        mul $t5, $t4, $t4               # Atualiza $t5 de acordo com o novo primo

        ble $t5, $s0, fatores_impares   # Se $s0 ainda for menor que o primo ao quadrado, faz a verificação novamente
```

Função que imprime o número atual ($s0), caso ele seja primo e não seja menor ou igual a 1. Após a impressão desvia para finalizar o programa.

```
se_primo:                               # Se o número atual for primo, ele é impresso como fator
        ble $s0,1, termina_programa     # Se o número for menor ou igual a 1, termina o programa

        li $v0, 1                       # Impressão de um inteiro
        la $a0, ($s0)                   # Imprime o número
        syscall                         # Faz a chamada do sistema

        li $v0, 4                       # Impressão de string
        la $a0, quebra_linha            # Imprime quebra de linha
        syscall                         # Faz a chamada do sistema

        j termina_programa              # Após a impressão, o programa pode ser finalizado
```

Função que trata as entradas inválidas, ou seja, números que não pertençam ao intervalo 1 < N < 32768.

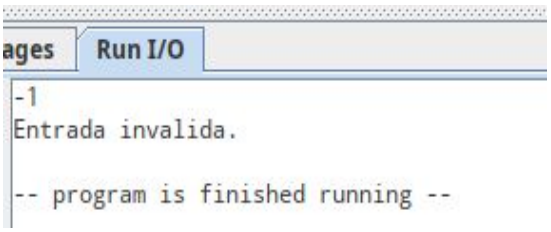
```
se_invalida:                            # Rotina que trata as entrandas inválidas
        li $v0, 4                       # Impressão de string
        la $a0, invalida                # Imprime "Entrada invalida.\n"
        syscall                         # Faz a chamada do sistema
```

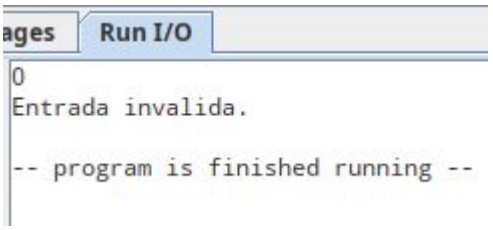
Função responsável por finalizar o programa.

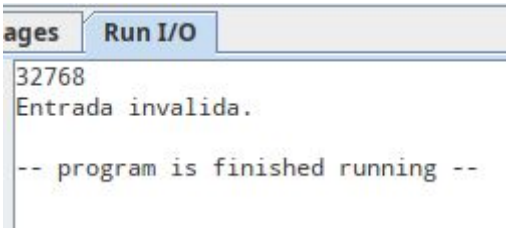
```
termina_programa:                       # Rotina para finalizar o programa
        li $v0, 10                      # Finaliza o programa
        syscall                         # Faz a chamada do sistema
```

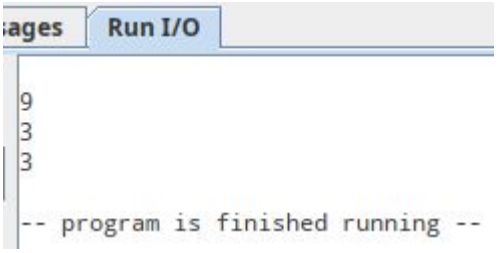
Execução

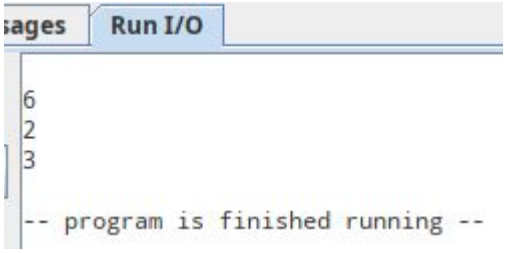
Para executar as instruções programas foi necessário montar o arquivo e então executá-lo no ambiente MARS.

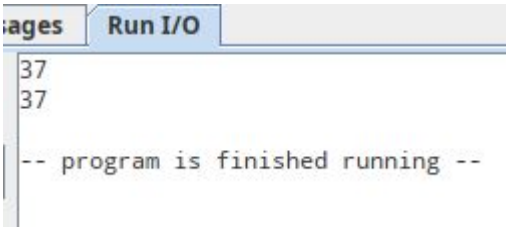


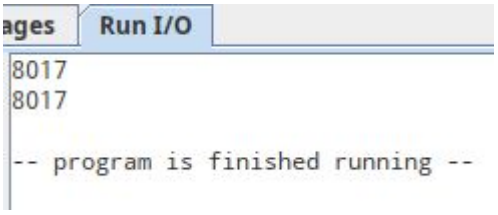












Limitações

A dupla encontrou dificuldades em imprimir os fatores corretos, pois o número precisava ser divido para fazer algumas verificações e era sempre usado o valor dividido. A solução para o problema foi copiar o valor para um registrador auxiliar ($t7) e fazer as divisões para verificação apenas no auxiliar, e quando necessário, atualizar o $s0. Além disso, para alguns números, o número 1 era impresso como fator. Foi verificado que estava sendo usada a instrução incorreta para o desvio e o erro foi corrigido e quando o número chega a 1, o programa é finalizado.

Além disso, o programa se limita apenas a fatorar inteiros positivos menores que 32768 e maiores que 1.