

Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Informática e Estatística
Ciência da Computação
INE5411 - Organização de Computadores I

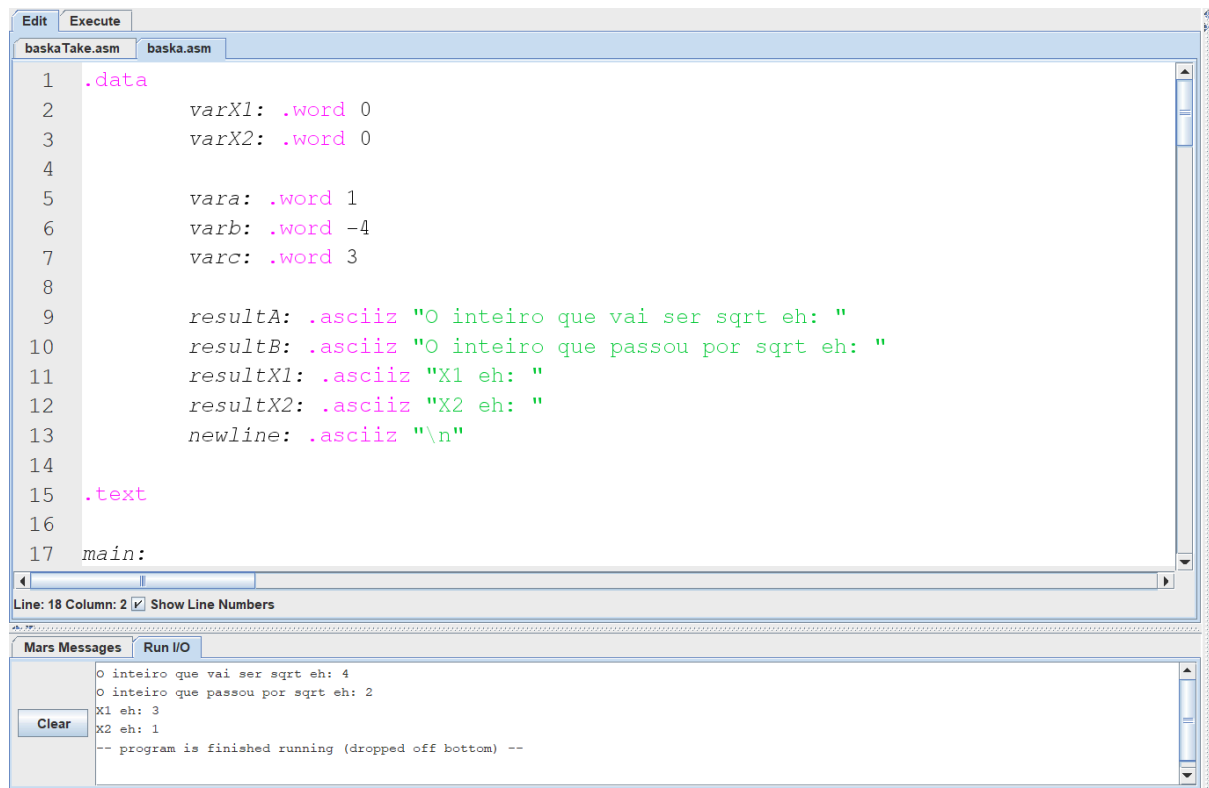
Relatório Laboratório 1

Joshua Cruz do Amaral (24205457)
Julia Macedo de Castro (23250860)

Florianópolis
2025

1. Atividades via Console

1.1. Fórmula de Bhaskara



The screenshot shows the Mars IDE with two tabs: 'baskaTake.asm' and 'baska.asm'. The 'baska.asm' tab is active, displaying the following assembly code:

```
1 .data
2     varX1: .word 0
3     varX2: .word 0
4
5     vara: .word 1
6     varb: .word -4
7     varc: .word 3
8
9     resultA: .asciiz "O inteiro que vai ser sqrt eh: "
10    resultB: .asciiz "O inteiro que passou por sqrt eh: "
11    resultX1: .asciiz "X1 eh: "
12    resultX2: .asciiz "X2 eh: "
13    newline: .asciiz "\n"
14
15 .text
16
17 main:
```

Below the code editor, the 'Mars Messages' and 'Run I/O' tabs are visible. The 'Run I/O' tab shows the output of the program:

```
O inteiro que vai ser sqrt eh: 4
O inteiro que passou por sqrt eh: 2
X1 eh: 3
X2 eh: 1
-- program is finished running (dropped off bottom) --
```

Acima segue imagem dos parâmetros iniciais ($a=1$, $b=-4$, $c=3$) e também alguns `.asciiz` usados para depuração. Após transferir os dados da memória para o registrador, converter o resultado de b^2-4ac para ponto flutuante em outro registrador específico para essa isso, daí então aplicar função de raiz quadrada, converter de volta para inteiro e seguir com a fórmula para então chegar em $X1$ e $X2$.

Como pode-se ver na print e na parte de RUN I/O:

O resultado de b^2-4ac é $((-4)^2 - 4 * 1 * 3) = 4$. Raiz de $4 = 2$.

Continuando a fórmula: $[- (-4) +/- 2] / 2 * 1$

$X1 = (4+2)/2 = 3$

$X2 = (4-2)/2 = 1$

Confirmando assim a acurácia do cálculo e resultado fornecido para os valores escolhidos.

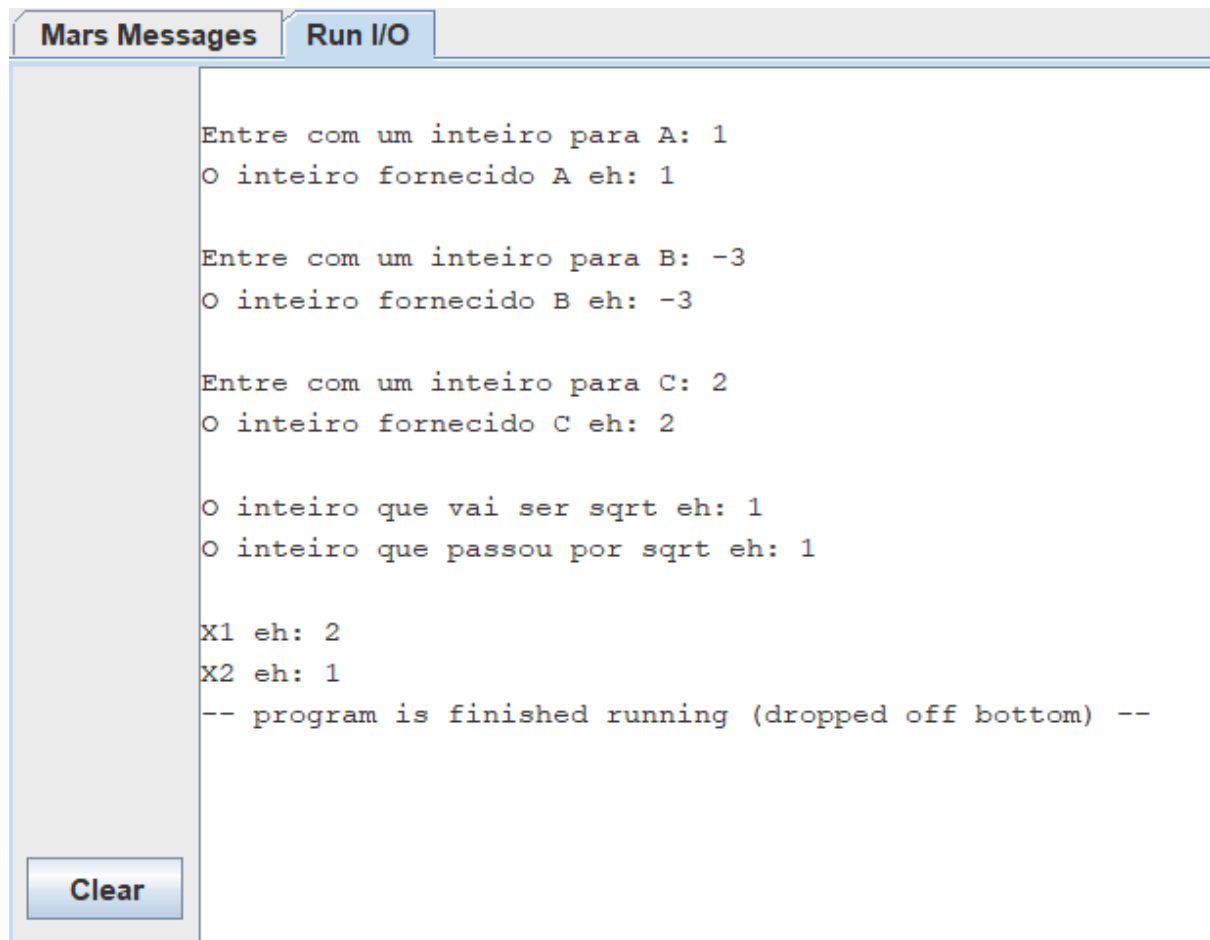
Também pode-se confirmar esse resultado (X1 e X2) nos registradores utilizados (\$s0 e \$s1):

Registers			Coproc 1	Coproc 0
Name	Number	Value		
\$zero	0	0x00000000		
\$at	1	0x10010000		
\$v0	2	0x00000001		
\$v1	3	0x00000000		
\$a0	4	0x00000001		
\$a1	5	0x00000000		
\$a2	6	0x00000000		
\$a3	7	0x00000000		
\$t0	8	0x00000002		
\$t1	9	0x00000006		
\$t2	10	0x00000002		
\$t3	11	0xffffffff4		
\$t4	12	0x00000004		
\$t5	13	0x00000004		
\$t6	14	0x00000001		
\$t7	15	0x00000002		
\$s0	16	0x00000003		
\$s1	17	0x00000001		
\$s2	18	0x00000000		
\$s3	19	0x00000000		
\$s4	20	0x00000000		
\$s5	21	0x00000000		
\$s6	22	0x00000000		
\$s7	23	0x00000000		
\$t8	24	0xffffffffc		
\$t9	25	0x00000003		
\$k0	26	0x00000000		
\$k1	27	0x00000000		
\$gp	28	0x10008000		
\$sp	29	0x7fffeffc		
\$fp	30	0x00000000		
\$ra	31	0x00000000		
pc		0x00400128		
hi		0x00000000		
lo		0x00000001		

1.2. Fórmula de Bhaskara com valores fornecidos pelo usuário

Não há diferença significativa do funcionamento do código em comparação com a questão anterior, a única distinção é que agora os valores de a,b,c são fornecidos pelo usuário.

Com a finalidade de verificar a continuidade da exatidão do cálculo com o mesmo código, números que já são conhecidos por resultarem em delta positivo e X1, X2 iguais a inteiros serão escolhidos.



Como pode-se ver na print e na parte de RUN I/O:

Foram escolhidos valores $a=1$, $b=-3$, $c=2$.

O resultado de b^2-4ac é $((-3)^2 - 4 * 1 * 2) = 1$. Raiz de $1 = 1$.

Continuando a fórmula: $[-(-3) +/- 1] / 2 * 1$

$X1 = (3+1)/2 = 2$

$X2 = (3-1)/2 = 1$

Confirmando assim a acurácia do cálculo e resultado fornecido para os valores escolhidos.

Também pode-se confirmar esse resultado ($X1$ e $X2$) nos registradores utilizados ($\$s0$ e $\$s1$):

Registers	Coproc 1	Coproc 0
Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x00000001
\$v1	3	0x00000000
\$a0	4	0x00000001
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000001
\$t1	9	0x00000004
\$t2	10	0x00000002
\$t3	11	0xffffffff8
\$t4	12	0x00000004
\$t5	13	0x00000003
\$t6	14	0x00000001
\$t7	15	0x00000002
\$s0	16	0x00000002
\$s1	17	0x00000001
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0xffffffffd
\$t9	25	0x00000002
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffeffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400208
hi		0x00000000
lo		0x00000001

Pergunta relacionada a 1.1 e 1.2:

*Existe diferença nas quantidades de linhas da coluna **Basic** e **Source**? Se sim, explique o motivo da diferença.*

R: Sim, há diferença do número de linhas. Em ambos os exercícios há mais linhas de código Basic do que Source, o que é explicado pelo fato do Mars fazer a conversão de pseudo-instruções (como *La* e *Sw*) em mais de uma instrução nativa, na qual o processador pode trabalhar.

Imagem de representação:

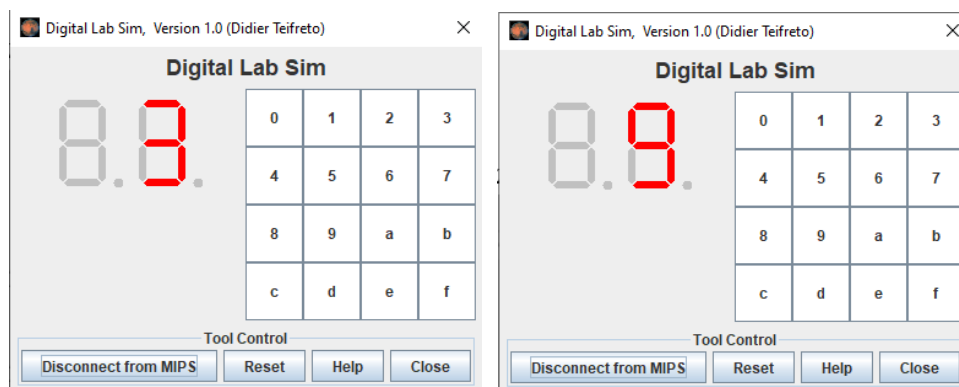
Edit		Execute		
Text Segment				
Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400180	0x15400001	bne \$10,\$0,0x00000001	174: div \$s1, \$t7, \$t2 # (-b + RA12 DE B QUADRADO - 4AC) / 2A
<input type="checkbox"/>	0x00400184	0x0000000d	break	
<input type="checkbox"/>	0x00400188	0x01ea001a	div \$15,\$10	
<input type="checkbox"/>	0x0040018c	0x00008812	mflo \$17	
<input type="checkbox"/>	0x00400190	0x24020004	addiu \$2,\$0,0x00000...	178: li \$v0,4
<input type="checkbox"/>	0x00400194	0x3c011001	lui \$1,0x00001001	179: la \$a0,newline
<input type="checkbox"/>	0x00400198	0x34240115	ori \$4,\$1,0x00000115	
<input type="checkbox"/>	0x0040019c	0x0000000c	syscall	180: syscall
<input type="checkbox"/>	0x004001a0	0x24020004	addiu \$2,\$0,0x00000...	183: li \$v0,4
<input type="checkbox"/>	0x004001a4	0x3c011001	lui \$1,0x00001001	184: la \$a0,newline
<input type="checkbox"/>	0x004001a8	0x34240115	ori \$4,\$1,0x00000115	
<input type="checkbox"/>	0x004001ac	0x0000000c	syscall	185: syscall
<input type="checkbox"/>	0x004001b0	0x24020004	addiu \$2,\$0,0x00000...	187: li \$v0,4
<input type="checkbox"/>	0x004001b4	0x3c011001	lui \$1,0x00001001	188: la \$a0,resultX1 #X1 :"
<input type="checkbox"/>	0x004001b8	0x34240105	ori \$4,\$1,0x00000105	
<input type="checkbox"/>	0x004001bc	0x0000000c	syscall	189: syscall
<input type="checkbox"/>	0x004001c0	0x24020001	addiu \$2,\$0,0x00000...	191: li \$v0,1
<input type="checkbox"/>	0x004001c4	0x00102021	addu \$4,\$0,\$16	192: move \$a0,\$s0
<input type="checkbox"/>	0x004001c8	0x0000000c	syscall	193: syscall
<input type="checkbox"/>	0x004001cc	0x3c011001	lui \$1,0x00001001	195: sw \$s0, varX1 #GUARDA VALOR X1 NA MEMÓRIA varX1
<input type="checkbox"/>	0x004001d0	0xac300000	sw \$16,0x00000000(\$1)	
<input type="checkbox"/>	0x004001d4	0x24020004	addiu \$2,\$0,0x00000...	198: li \$v0,4

Observação: foi notado que não há tratamento para casos especiais que podem/vão causar exceções, como quando o resultado de Delta é negativo ou quando $a=0$. Porém levando em consideração o escopo desse projeto e por verificações como essa não terem sido mencionadas nas especificações deste Laboratório, o time decidiu por não implementá-las.

2. Atividades via Digital Lab

2.1. Números em sequência no display

O endereço do display é carregado no registrador \$s0 (0xFFFF0010), correspondente ao display da direita. Depois, o programa entra em um loop infinito onde são carregados cada um dos valores hexadecimais (que se referem aos padrões de segmentos acesos para cada dígito de 0 a 9) no registrador \$t0. Cada valor é armazenado no endereço do display por meio da instrução *sb* (store byte), o que atualiza a visualização do display para o dígito correspondente. Foi utilizado *sb* ao invés de *sw* (store word), pois o display recebe informações byte a byte, não sendo necessário armazenar uma palavra inteira na memória. Quando chega no dígito 9, o programa retorna ao início do loop, repetindo a sequência.



2.2. Teclas no display

Inicialmente, os endereços correspondentes à ativação da linha do teclado, leitura do valor da tecla pressionada e display de sete segmentos são carregados em registradores. As tabelas com os códigos das teclas (em “teclas”) e seus respectivos padrões binários para exibição no display (em “segmentos”) são posicionadas na memória e seus endereços são guardados em registradores.

O programa entra no loop principal, onde é feita uma varredura iterativamente das linhas do teclado ativando uma de cada vez (com valores 1, 2, 4 e 8) para identificar qual tecla foi pressionada (se alguma for). Para cada linha ativada, lê o código da tecla pressionada. Se nenhuma tecla estiver ativa, passa para a próxima linha. Ao detectar uma tecla pressionada, o programa procura o código lido na tabela de códigos das teclas para localizar o índice correspondente. Esse índice é usado para acessar o padrão binário correto na tabela “segmentos”, que é então escrito no endereço do display para mostrar a tecla pressionada. Esse procedimento se repete continuamente, garantindo que a qualquer momento o display representa a última tecla clicada.

