

Revisão P1

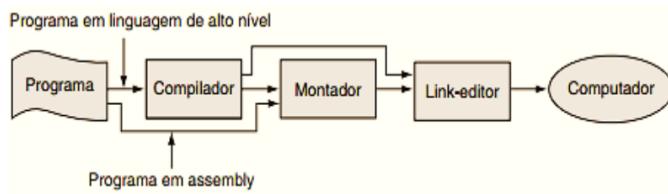
▼ Questão 1

(F) É possível afirmar que os compiladores são usados para gerar o executável a partir do programa objeto?

O papel dos compiladores é de converter o código de linguagem alto nível em programa objeto. A etapa de criação do executável é realizada na “linkagem”, processo que será feita a associação de bibliotecas e outros elementos específicos do OS junto ao programa objeto, para que seja executável naquela máquina.

Alguns compiladores produzem código Assembly que é passado para um montador para posterior processamento (Montagem e Link-edição).

Alguns compiladores produzem o trabalho dos montadores sem gerar o código Assembly;



(F) Compilação Cruzada é utilizada somente quando se tem O.S.s iguais, mas em versões diferentes

Compilação Cruzada é o caso em que desenvolve-se um código para uma determinada máquina alvo que possui arquitetura diferente da máquina local.

(F) O código objeto se difere do código executável apenas pela etapa de montagem

O código objeto é apenas o código em binário, já o executável é o mesmo código, porém já “linkado”.

(F) Os programas executáveis gerados em dois computadores idênticos, porém com O.S diferente, a partir do mesmo código fonte em baixo nível, serão sempre iguais.

Ao passarem pelo processo de link, os códigos objetos serão diferentes, pois cada O.S. terá suas próprias dependências e bibliotecas associadas.

▼ Questão 2

Complete a tabela:

```
.data 0x10010004
var1: .half 0xd # 2 bytes
var2: .word 0x15 # 4 bytes
var3: .ascii "CAFE" # 1 byte cada caracter
```

ENDEREÇO	DADOS
0x10010000	
0x10010004	0x0000000d
0x10010008	0x00000015
0x1001000C	CAFE → 0x45464143

▼ Questão 3

REGISTRADOR	
ENDEREÇO	DADO
\$t0	0x10010001
\$t1	0x10010004
\$t2	0x00000030
\$t3	0x00000040
\$t4	0x00000000
\$t5	0x00000000
\$t6	0xABCD EFOO
\$t7	0x00000000
\$s0	0x00000000
\$s1	0x00000050
\$s2	0x00000060
\$s3	0x00000000

MEMÓRIA	
ENDEREÇO	DADO
0x10010000	0xAA
0x10010001	0x1C
0x10010002	0x53
0x10010003	0x28
0x10010004	0x84
0x10010005	0xF1
0x10010006	0x12
0x10010007	0x64
0x10010008	0xE5
0x10010009	0x87
0x1001000A	0x99
0x1001000B	0x3D

▼ a)

```
lh $t2, 4($t1)
```

\$t1 = 0x84

4(\$t1) = 0xE5

\$t2 = 0xE5

▼ b)

```
sb $t3, 8($t0)
```

\$t3 = 0x00000040

\$t0 = 0x10010001

8(\$t0) = 0x10010009

0x10010009 = 0x00000040

▼ Questão 4

Escreva uma instrução em Assembly MIPS para receber 3 valores inteiros e retornar se a soma delas é menor, igual ou maior que 100

```
.data
    input_str_1: .asciiz "Primeiro valor: "
    input_str_2: .asciiz "Segundo valor: "
    input_str_3: .asciiz "Terceiro valor: "

    output_str_1: .asciiz "Menor que 100"
    output_str_2: .asciiz "Igual a 100"
    output_str_3: .asciiz "Maior que 100"

.text
    # Recebendo primeiro número
    li $v0, 4
    la $a0, input_str_1
    syscall

    li $v0, 5
    syscall

    add $t0, $v0, $0
```

```

# Recebendo segundo número
li $v0, 4
la $a0, input_str_2
syscall

li $v0, 5
syscall

add $t1, $v0, $0

# Recebendo terceiro número
li $v0, 4
la $a0, input_str_3
syscall

li $v0, 5
syscall

add $t2, $v0, $0

# Soma inputs
add $t3, $t0, $1 # a + b
add $t3, $t3, $t2 # (a + b) + c

blt $t3, 100, less
bgt $t3, 100, greater
li $v0, 4
la $a0, output_str_2
syscall
j exit

greater:
li $v0, 4
la $a0, output_str_3
syscall
j exit

less:
li $v0, 4
la $a0, output_str_1
syscall

exit:

```

▼ Questão 5

Quais as etapas necessárias à geração de um código executável a partir de um código fonte escrito em linguagem de baixo nível? Apresente-as na sequência.

Ao se escrever um código em linguagem de baixo nível, é primeiramente feita a montagem, que gerará o Código Objeto. Logo após, acontece o processo de

“link”, que serão adicionadas as dependências necessárias ao Código Objeto, como bibliotecas e outros componentes de execução do próprio O.S. Assim, é criado o executável.

▼ Questão 6

Com relação as arquiteturas RISC e CISC dos processadores, em nível de conjunto de instruções, são feitas as seguintes afirmações:

(V) Cada instrução RISC é decodificada sem a necessidade de um microcódigo, ou seja, via hardware.

(F) O tempo de execução de uma sequência de instrução é maior na arquitetura RISC uma vez que as instruções são complexas.

RISC

- Pouco número de instruções, **mais difícil e maior escrita** →
Ocupa mais memória
- Endereçamento de instruções por registradores, através de LOAD e STORE.
- Decodificação por **hardware** (Mais rápido)
- OP-Codes maiores, menores ciclos de execução para o processador

CISC

- Grande número de instruções, **mais fácil de escrita**
- Endereçamento de instruções por **microcódigo**
- Interpretação do op-code por **software** (Mais lento)
- OP-Codes menores, pois tem muitas **instruções diversificadas**

▼ Questão 7

Como sabemos, no set de instruções do MIPS há instruções básicas e pseudo-instruções. Esta última não está implementada em hardware. Por este motivo são substituídas por instruções

básicas no processo de montagem. Cite um exemplo de pseudo-instrução e sua instrução básica equivalente.

A instrução *move*, por exemplo. Seu equivalente seria a operação de *add*.

```
move $t1, $t2  
add $t1, $t2, $0
```

▼ Questão 8

Trascreva o seguinte trecho de código para uma linguagem de alto nível

Considerando que $a = \$t1$, $b = \$t2$, $c = \$t3$, $d = \$t4$ e $x = \$t5$:

```
.text  
  
li $t1, 1  
li $t2, 2  
li $t3, 3  
li $t4, 4  
li $t5, 10  
blt $t2,$t1, Exit  
bgt $t3,$t4,c2  
addi $t5, $t5, -1  
j Exit  
c2: addi $t5,$t5,1  
Exit:
```

```
int a = 1  
int b = 2  
int c = 3  
int d = 4  
int x = 10  
  
if(b < a){  
    exit();  
}  
else{  
    if(c > d){  
        x ++;  
        exit();  
    }  
    else{  
        x --;  
        exit();  
    }  
}
```

▼ Questão 9

```

.data
a: .half 7,10
b: .byte 5
c: .byte 50
d: .word 0x86
e: .byte 0x90
f: .ascii "C63S"
g: .word 15
h: .half 14,15
i: .byte 8

```

Endereço	Dado
0x10010000	0x000a0007
0x10010004	0x00003205
0x10010008	0x00000086
0x1001000C	0x00C63s90
0x10010010	0x00000015
0x10010014	0x00140015
0x10010018	0x00000008
0x1001001C	0x00000000

▼ Questão 10

Registradores

Endereço	Dado
\$t0	0x10010000
\$t1	0x10010008
\$t2	0x00000008
\$t3	0x10010006
\$t4	0x000000A3
\$t5	0x00000096
\$t6	0x10010004
\$t7	0x00000230
\$s0	0x0000006B
\$s1	0x000000eb
\$s2	0x1001000A
\$s3	0xa83fc12e
\$s4	0x00000001

Memória

Endereço	Dado
0x10010000	0x4D
0x10010001	0x94
0x10010002	0x78
0x10010003	0x2C
0x10010004	0x7E
0x10010005	0XAA
0x10010006	0x50
0x10010007	0x89
0x10010008	0x15
0x10010009	0x5E
0x1001000A	0XF8
0x1001000B	0x9C
0x1001000C	0x8A

▼ a) sw \$s3, 8(\$t0)

Registradores

Endereço	Dado

Memória

Endereço	Dado

\$t0	0x10010000
\$t1	0x10010008
\$t2	0x00000008
\$t3	0x10010006
\$t4	0x000000A3
\$t5	0x00000096
\$t6	0x10010004
\$t7	0x00000230
\$s0	0x0000006B
\$s1	0x000000eb
\$s2	0x1001000A
\$s3	0xa83fc12e
\$s4	0x00000001

0x10010000	0x4D
0x10010001	0x94
0x10010002	0x78
0x10010003	0x2C
0x10010004	0x7E
0x10010005	0XAA
0x10010006	0x50
0x10010007	0x89
0x10010008	0xa83fc12e
0x10010009	0x5E
0x1001000A	0XF8
0x1001000B	0x9C
0x1001000C	0x8A

▼ b) lw \$t2, 4(\$t6)

Registradores

Endereço	Dado
\$t0	0x10010000
\$t1	0x10010008
\$t2	0x15
\$t3	0x10010006
\$t4	0x000000A3
\$t5	0x00000096
\$t6	0x10010004
\$t7	0x00000230
\$s0	0x0000006B
\$s1	0x000000eb
\$s2	0x1001000A
\$s3	0xa83fc12e
\$s4	0x00000001

Memória

Endereço	Dado
0x10010000	0x4D
0x10010001	0x94
0x10010002	0x78
0x10010003	0x2C
0x10010004	0x7E
0x10010005	0XAA
0x10010006	0x50
0x10010007	0x89
0x10010008	0x15
0x10010009	0x5E
0x1001000A	0XF8
0x1001000B	0x9C
0x1001000C	0x8A

▼ c) lh \$t5, 6(\$t0)

Registradores

Memória

Endereço	Dado
\$t0	0x10010000
\$t1	0x10010008
\$t2	0x00000008
\$t3	0x10010006
\$t4	0x000000A3
\$t5	0x50
\$t6	0x10010004
\$t7	0x00000230
\$s0	0x0000006B
\$s1	0x000000eb
\$s2	0x1001000A
\$s3	0xa83fc12e
\$s4	0x00000001

Endereço	Dado
0x10010000	0x4D
0x10010001	0x94
0x10010002	0x78
0x10010003	0x2C
0x10010004	0x7E
0x10010005	0XAA
0x10010006	0x50
0x10010007	0x89
0x10010008	0x15
0x10010009	0x5E
0x1001000A	0XF8
0x1001000B	0x9C
0x1001000C	0x8A

▼ d) sb \$s0, 0(\$t3)

Registradores

Endereço	Dado
\$t0	0x10010000
\$t1	0x10010008
\$t2	0x00000008
\$t3	0x10010006
\$t4	0x000000A3
\$t5	0x00000096
\$t6	0x10010004
\$t7	0x00000230
\$s0	0x0000006B
\$s1	0x000000eb
\$s2	0x1001000A
\$s3	0xa83fc12e
\$s4	0x00000001

Memória

Endereço	Dado
0x10010000	0x4D
0x10010001	0x94
0x10010002	0x78
0x10010003	0x2C
0x10010004	0x7E
0x10010005	0XAA
0x10010006	0x0000006B
0x10010007	0x89
0x10010008	0x15
0x10010009	0x5E
0x1001000A	0XF8
0x1001000B	0x9C
0x1001000C	0x8A