



Nome: GABARITO

Matrícula: 18/0123456

d₀ d₁ / d₂ d₃ d₄ d₅ d₆ d₇ d₈

Prova 1

(3.0) 1) Dado o programa em C ao lado.

(1.0) a) Respeitando a convenção do uso de registradores, compile eficientemente para Assembly RISC-V ISA RV32I.

(1.0) b) Para um processador RISC-V, onde as instruções de acesso à memória possuem CPI=2 e todas as outras CPI=1, com frequência de *clock* de 1GHz, qual a CPI média obtida para este programa? 1

Qual o tempo necessário à sua execução? 64 ns

Obs.: Considere `ecall` como uma instrução com CPI=1.

(1.0) c) Explique por que a observação feita é importante para a resolução do item b).

```
int mdc(int i, int j){  
    while (i != j)  
        if (i > j)  
            i = i - j;  
        else  
            j = j - i;  
    return i;  
}  
  
void main(){  
    int a, i=d6d7d8, j=d3d4d5;  
    a = mdc(i, j);  
    printf("mdc=%d\n", a);  
}
```

(3.0) 2) Considere o procedimento abaixo, onde o processador RISC-V está configurado para o modo de arredondamento ao mais próximo ($rm=111$), com os argumentos $fa0=0x43000000$ e $fa1=0x3f800000$.

PROC: `fsqrt.s fa0, fa0`

<u>0x58057553</u>	<u>0,2</u>
<u>0xA0B52553</u>	<u>0,7</u>
<u>0xFE050CE3</u>	<u>0,2</u>
<u>0X00256513</u>	<u>0,2</u>
<u>0X00008067</u>	<u>0,2</u>

`feq.s a0, fa0, fa1`
`beq a0, zero, PROC`
`ori a0, a0, 2`
`ret jalr zero, RA, 0`

(1.0)a) Escreva o código em linguagem de máquina, **em hexadecimal**, ao lado de cada instrução.

(1.0)b) O valor final em IEEE754 **em hexadecimal** contido registrador fa0 é 0x3F800000

(1.0)c) Se o valor retornado no registrador a0 for tratado como uma instrução da ISA RISC-V, qual o mnemônico completo desta instrução? LB ZERD, 0(ZERO)

(2.5) 3) A ISA ARMv7 é muito mais flexível que a ISA RISC-V, aproximando aos processadores CISC. Entre as instruções complexas daquela ISA estão as instruções de backup do banco de registradores na pilha, geralmente realizado no início de um procedimento, e a sua restauração com retorno do procedimento. Considerando que esta funcionalidade é importante em uma ISA, implemente as seguintes pseudoinstruções:

(1.0) a) `STMFD #` empilha os registradores x1 a x31 a partir endereço apontado por sp atualizando-o.

(1.5) b) `LDMFD #` restaura os registradores x1 a x31 a partir do endereço apontado por sp atualizando-o e retorna para o endereço apontado por ra.

(2.5) 4) Responda:

(1.5) a) Explique i) o que é, ii) onde se localiza e iii) para que serve o *chipset* de um computador.

(1.0) b) “Um processador capaz de executar 10.000.000 de instruções por segundo é mais eficiente que outro capaz de executar 5.000.000 de instruções por segundo.” A afirmação é verdadeira ou falsa? Por que?

1º PROVA
GABARITO

1)

- a) Ver listagem em Anexo 1,0
- b) No Programa não há instrução explícita de acesso à memória (lw ou sw).

Os acessos são feitos pela instrução `ECALL` que foi dito ter $CPI = 1$.

Logo:

$$CPI_{mem} = 1 \quad (0,5)$$

$$t_{exec} = I \times CPI \times T$$

Para $i=456$ e $j=123$, são executadas 64 instruções.

Logo

$$t_{exec} = 64 \times 1 \times \frac{1}{1 \times 10^9} = 64 \text{ ns} \quad (0,9)$$

c) Sendo `ECALL` uma instrução que chama serviços do sistema operacional, a CPI real depende da rotina de sistema chamada que não é de nosso conhecimento.

1,0

2)

a) na folha da paoxa

b) $FAO = 0x43000000 \rightarrow 128,0$

$FA1 = 0x3F800000 \rightarrow 1,0$

PROC: FSGRTS FAO, FAO

FEGTS AD, FAO, FA1

beg AD, ZERO, PROC

FICA NO LOOP

ATE FAO SE REDUZIR

a 1,0

Logo VALOR FINAL FAO = FA1

1,0

AD = 0x0000 0001 AD final do LOOP

Logo AD = 0...0001 OR 0...0010 = 00...0011 = 3
2

AD = 0x0000 0003

↳ LB ZERO, 0(ZERO)

1,0

3) a) STMFD

→ não precisa salvar X0 na pilha

addi SP, SP, -124

SW X1, 0(SP)

SW X2, 4(SP)

SW X3, 8(SP)

SW X31, -128(SP)

SW X30, 116(SP) → Precisa de 6 mem

SW X31, 120(SP)

OPERAÇÃO DE SP

1,0

b) UDMFD

LW X1, 0(C\$P)

LW X2, 4(C\$P)

LW X3, 8(C\$P)

(1,0)

LW X30, 116(C\$P)

LW X31, 120(C\$P)

addi SP, SP, 124

jalR ZERO, RA, 0

(0,5)

4)

a) Os chips set correspondem aos circuitos que controlam o acesso do processador aos dispositivos de I/O. Se localizam na placa-mãe nos computadores PC correspondem à parte norte e parte sul.

(1,5)

b) Falsa, para a eficiência depende da capacidade das instruções, além de poder depender de outros fatores, como consumo, tamanho, etc.

(1,0)

```
.eqv IX 456  
.eqv JX 123
```

```
.data
```

```
L1: .string "mdc="  
L2: .string "\n"
```

```
.text
```

```
MAIN: li a0, IX  
      li a1, JX  
      call MDC  
      mv t0, a0
```

} 0, 2

```
la a0, L1  
li a7, 4  
ecall
```

} 0, 2

```
mv a0, t0  
li a7, 1  
ecall
```

} 0, 2