Universidade de Brasília
Departamento de Ciência da Computação

Disciplina: CIC 116394 – Organização e Arquitetura de Computadores – Turma A 2015/2
Prof. Marcus Vinicius Lamar

d_0 d_1 / d_2 d_3 d_4 d_5 d_6 d_7 d_8

Nome: GABARITO Matrícula: 1 2 / 3 4 5 6 7 8 9

Prova 1

(3.0)1) Dado o programa main em C ao lado:

(2.0)a) Compile-o para Assembly MIPS, respeitando a convenção do uso dos registradores.

(0.5)b) Qual o tamanho em bytes o seu código compilado ocupa na memória RAM, considerando código e dados?

(0.5)c) Para i=1 qual o número de instruções executadas pelo seu código? SQRT

```
void main(void) {
   float i;
   scanf("%f",&i);
   printf("%f", Sqrt(i,1.0)); }

float Sqrt(float num, float prev) {
   float next = (prev+num/prev)/2;

   if (fabs(next-prev)<1e-10*next)
      return next;

   return Sqrt(num, next); }
```

(5.0)2) Processadores com palavras de apenas 8 bits são suficientes para uma ampla variedade de aplicações. A implementação e uso de instruções SIMD (*Single Instruction, Multiple Data*) vem sendo cada vez mais popular, principalmente em aplicações gráficas. Você é responsável pela adaptação da ISA de um processador MIPS32 para criação de um novo conjunto de pseudo-instruções SIMD com palavras de 8 bits que seja capaz de realizar operações lógicas e aritméticas. Considere que cada registrador da ISA MIPS armazene um conjunto de 4 números de 8 bits, cada número representado em complemento de 2. Realize implementações otimizadas das pseudo-instruções abaixo desta nova ISA MIPS SIMD:

(0.5)a) lbv $t0, Imm($t1) # Carrega em $t0 4 números de 8 bits a partir dos endereços ($t1+Imm), ($t1+Imm+1), ($t1+Imm+2) e ($t1+Imm+3) da memória

(1.0)b) sbvi n, $t0, Imm($t1) # Armazena no endereço ($t1+Imm) o n-ésimo número (n=2 bits, de 0 a 3) do registrador $t0 Dica: considere n=0 o número no localizado no LSB e n=3 o número no MSB

(0.5)c) andv $t0, $t1, $t2 # Armazena em $t0 o resultado da operação and bit a bit dos 4 números em $t2 e $t1

(1.0)d) addv $t0, $t1, $t2 # Armazena em $t0 os resultados das somas dos 4 números em $t1 com os 4 números em $t2, com detecção de overflow, caso ocorra em qualquer das operações

(1.0)e) multv $t0, $t1, $t2 # armazena em $t0 os 8 bits menos significativos dos resultados das multiplicações dos 4 números em $t1 com os 4 números em $t2, sem detecção de overflow

(1.0)f) cvi n, $t0, $t1 # converte o n-ésimo número armazenado em $t1 para um número de 32 bits em $t0

(3.0)3) Considere os gráficos e tabelas apresentados (verso) por Blem *et.al.* no artigo "*ISA Wars: Understanding the Relevance of ISA being RISC or CISC to Performance, Power, and Energy on Modern Architectures*", publicado na *ACM Transactions on Computer Systems*, Vol. 33, No. 1, Article 3, em Março de 2015.

(0.5)a) No artigo, os autores utilizam vários benchmarks para tecer suas observações. Qual a diferença fundamental entre o SPEC INT e o SPEC FP?

(1.0)b) Como podemos observar, pelos dados mostrados, os achados 1, 2 e 3 apresentados na Tabela 2?

(1.0)c) Como podemos observar, pelos dados mostrados, os achados 7, 8 e 10 apresentados na Tabela 2?

(0.5)d) Segundo os autores, que tipo de arquitetura é mais eficiente energeticamente, RISC ou CISC? Que característica do processador mais afeta a sua eficiência energética, segundo este estudo?

BOA SORTE!

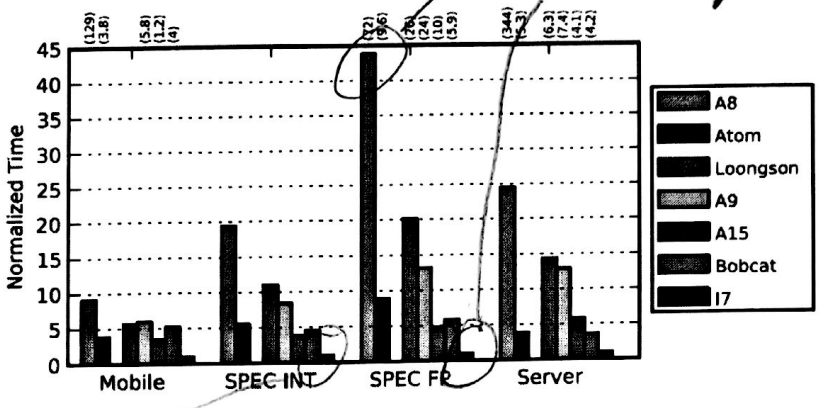

Fig. 2. Execution time normalized to i7.

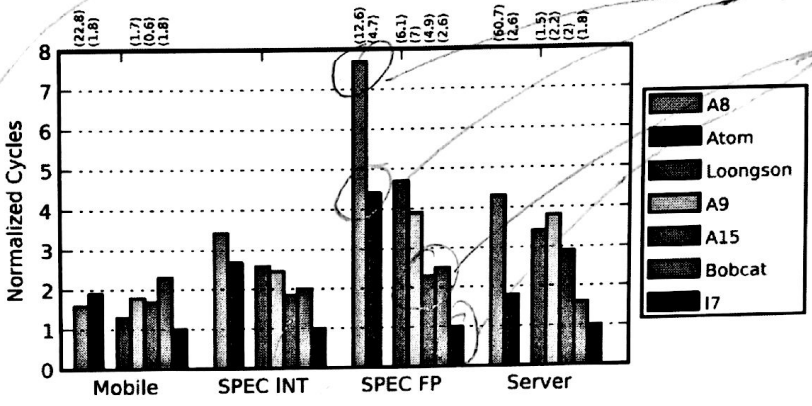

Fig. 3. Cycle count normalized to i7.

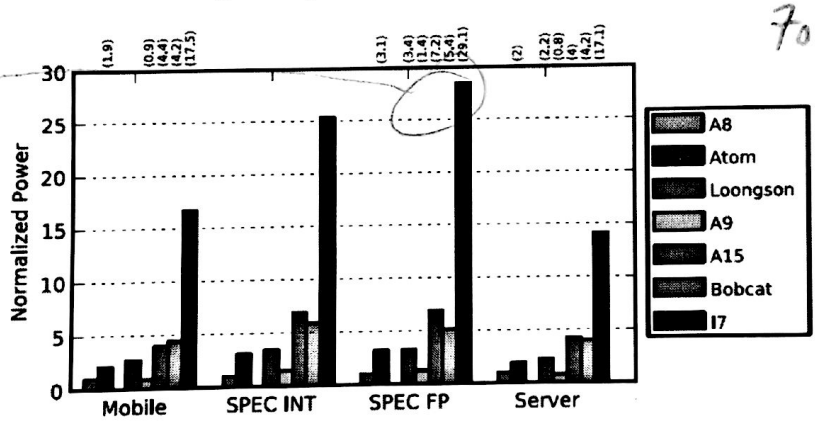

Fig. 8. Raw average power normalized to A8.

Table 2: Findings of VRG RISC vs CISC study

Performance,
1. Large performance gaps exist.
2. Cycle-count gaps are less than 3× (A8 to Atom OoO Processors to i7)
3. Cycles per instruction (CPI) can be less on x86 implementation
4. ISA performance effects indistinguishable between x86 and ARM
5. Microarchitecture, not the ISA, responsible for performance differences
6. Beyond micro-op translation, x86 ISA introduces no overheads over ARM and MIPS ISA
Power
7. Power consumption does not have a direct correlation to ISA
8. Choice of power or perf. optimization impacts power use more than does ISA
9. Energy use primarily a design choice; ISA's impact insignificant
Tradeoffs
10. High-performance processors require more power than lower-performance processors
11. It is the microarchitecture and design methodology that really matters

Table VI. Geometric Mean CPI a cross All Benchmark Suites

ISA	ARM	x86	MIPS	ARM	ARM	x86	x86
Implementation	A8	Atom	Loongson	A9	A15	Bobcat	i7
CPI	2.5	1.9	1.5	1.6	1.2	1.3	0.7

Table VII. Instruction Size Summary

		(a) Binary Size (MB)			(b) Instruction Length (B)		
		MIPS	ARM	x86	MIPS	ARM	x86
Mobile	Minimum	–	0.02	0.02	4.0	4.0	2.4
	Average	0.55	0.95	0.87	4.0	4.0	3.3
	Maximum	–	1.30	1.42	4.0	4.0	3.7
Desktop INT	Minimum	0.61	0.53	0.65	4.0	4.0	2.7
	Average	1.58	1.47	1.46	4.0	4.0	3.1
	Maximum	4.35	3.88	4.05	4.0	4.0	3.5
Desktop FP	Minimum	0.76	0.66	0.74	4.0	4.0	2.6
	Average	1.81	1.70	1.73	4.0	4.0	3.4
	Maximum	5.21	4.75	5.24	4.0	4.0	6.4
Server	Minimum	0.16	0.12	0.18	4.0	4.0	2.5
	Average	0.51	0.39	0.59	4.0	4.0	3.2
	Maximum	0.84	0.47	1.00	4.0	4.0	3.7

OAC-A 1ª PROVA 2015/2
GABARITO

1) a) EM ANEXO

b) Tamanho = 3x4 + 31x4 = 136 0,5
.DATA .TEXT

La = LUi + ORi =

c) Função SYRI ci i=1 ⇒ 19 instruções 0,5

2) em Anexo
a)
b)
c)
d)
e)

3)
a) SPEC INT → Benchmark de inteiros 0,5
SPEC FP → Benchmark de Float Point

b) na Folha
c) na Folha

d) Segundo o item 11, RISC ou CISC não implica A microArquitetura (organização) e a metodologia de Projeto são as características que mais afetam 0,5

```
1   .data
2   E10: .float 1e-10
3   UM: .float 1.0
4   DOIS: .float 2.0
5
6   .text
7       li $v0,6
8       syscall # scanf   retorno em $f0
9
10      la $t0,UM
11      lwc1 $f1,0($t0)   # carrega 1.0
12
13      jal Sqrt   #argumentos $f0 e $f1
14             # retorno em $f12
15      li $v0,2   # printf
16      syscall
17
18      li $v0,10  #exit
19      syscall
20
21  Sqrt:   addi $sp,$sp,-4
22      sw $ra,0($sp)       # salva $ra
23
24      la $t0,DOIS     # carrega 2.0
25      lwc1 $f10,0($t0)
26
27      div.s $f12,$f0,$f1
28      add.s $f12,$f12,$f1
29      div.s $f12,$f12,$f10    #retorno em $f12
30
31      sub.s  $f11,$f12,$f1
32      abs.s $f11,$f11
33
34      la $t0,E10      #carrega 1e-10
35      lwc1 $f9,0($t0)
36
37      mul.s $f9,$f9,$f12
38      c.lt.s 0,$f11,$f9
39      bc1t 0, SAIR
40
41      mov.s $f1,$f12
42      jal Sqrt
43
44  SAIR:   lw $ra,0($sp)       # retorno em $f12
45      addi $sp,$sp,4
46
47      jr $ra
48
```

```
.data
TESTE: .word 0x12345678

.text
    la $t1,TESTE
# a) lbv $t0,Imm($t1)
A:  lw $t0,0($t1)

    la $t0,0x12345678
    la $t1,TESTE
# b) sbvi n,$t0,Imm($t1)
B:  addi $at,$zero, 1
    sll $at,$at,3
    srlv $at,$t0,$at
    sb $at,0($t1)

    la $t1,0x12345678
    la $t2,0xF0F0F0F0
# c) andv $t0,$t1,$t2
C:  and $t0,$t1,$t2

    la $t1,0x12345678
    la $t2,0x01010101
# d) addv $t0,$t1,$t2
D:  addi $sp,$sp,-8
    sw $t3,0($sp)
    sw $t4,4($sp)

    andi $t3,$t1,0x00FF
    sll $t3,$t3,24
    andi $t4,$t2,0x00FF
    sll $t4,$t4,24
    add $at,$t3,$t4
    srl $at,$at,24
    or $t0,$zero,$at

    andi $t3,$t1,0xFF00
    sll $t3,$t3,16
    andi $t4,$t2,0xFF00
    sll $t4,$t4,16
    add $at,$t3,$t4
    srl $at,$at,16
    or $t0,$t0,$at

    lui $at,0x00FF
    and $t3,$t1,$at
    sll $t3,$t3,8
    and $t4,$t2,$at
    sll $t4,$t4,8
    add $at,$t3,$t4
    srl $at,$at,8
    or $t0,$t0,$at

    lui $at,0xFF00
    and $t3,$t1,$at
    and $t4,$t2,$at
    add $at,$t3,$t4
    or $t0,$t0,$at
```

```
59
60      lw $t3,0($sp)
61      lw $t4,4($sp)
62      addiu $sp,$sp,8
63
64
65      la $t1,0x03030303
66      la $t2,0x01020304
67  # e) multv $t0,$t1,$t2
68  E:  addi $sp,$sp,-8
69      sw $t3,0($sp)
70      sw $t4,4($sp)
71
72      andi $t3,$t1,0x00FF
73      andi $t4,$t2,0x00FF
74      mul $at,$t3,$t4
75      andi $t0,$at,0x00FF
76
77      andi $t3,$t1,0xFF00
78      srl $t3,$t3,8
79      andi $t4,$t2,0xFF00
80      srl $t4,$t4,8
81      mul $at,$t3,$t4
82      sll $at,$at,8
83      or $t0,$t0,$at
84
85      lui $at,0x00FF
86      and $t3,$t1,$at
87      srl $t3,$t3,16
88      and $t4,$t2,$at
89      srl $t4,$t4,16
90      mul $at,$t3,$t4
91      sll $at,$at,16
92      or $t0,$t0,$at
93
94      lui $at,0xFF00
95      and $t3,$t1,$at
96      srl $t3,$t3,24
97      and $t4,$t2,$at
98      srl $t4,$t4,24
99      mul $at,$t3,$t4
100     sll $at,$at,24
101     or $t0,$t0,$at
102
103     lw $t3,0($sp)
104     lw $t4,4($sp)
105     addiu $sp,$sp,8
106
107
108     la $t1,0x12845678
109 # f) cvi n,$t0,$t1
110 F:  addi $at,$zero,2
111     sll $at,$at,3
112     srlv $at,$t1,$at
113     addi $sp,$sp,-1
114     sb $at,0($sp)
115     lb $t0,0($sp)
116     addi $sp,$sp,1
```