Text Segment

0x00400000 0x3c0800f3 lui $8,0x000000f3 17 main: lui $t0,0xf3 #

0x00400004 0x35080023 ori $8,$8,0x00000023 18 ori $t0,$t0,0x23 # $t0<= 0x00f30023

0x00400008 0x3c090052 lui $9,0x00000052 19 lui $t1,0x52 #

0x0040000c 0x352900e2 ori $9,$9,0x000000e2 20 ori $t1,$t1,0xe2 # $t1<= 0x005200e2

0x00400010 0x3c0a0000 lui $10,0x00000000 21 lui $t2,0x00 #

0x00400014 0x354a008f ori $10,$10,0x0000008f22 ori $t2,$t2,0x8f # $t2<= 0x0000008f

0x00400018 0x112a002e beq $9,$10,0x0000002e 23 beq $t1,$t2,loop # Obviamente, esta instru��o nunca deve saltar

0x0040001c 0x00000000 nop 24 nop # Não faz nada mesmo

0x00400020 0x152a0001 bne $9,$10,0x00000001 25 bne $t1,$t2,next\_i # Obviamente, esta instru��o sempre deve saltar

0x00400024 0x254a008f addiu $10,$10,0x00000026 addiu $t2,$t2,0x8f # Obviamente, esta instru��o nunca deve executar

0x00400028 0x01095821 addu $11,$8,$9 27 next\_i: addu $t3,$t0,$t1 # $t3<= 0x00f30023 + 0x005200e2 = 0x01450105

0x0040002c 0x01096023 subu $12,$8,$9 28 subu $t4,$t0,$t1 # $t4<= 0x00f30023 - 0x005200e2 = 0x00a0ff41

0x00400030 0x01296823 subu $13,$9,$9 29 subu $t5,$t1,$t1 # $t5<= 0x0

0x00400034 0x01097024 and $14,$8,$9 30 and $t6,$t0,$t1 # $t6<= 0x00f30023 and 0x005200e2 = 0x00520022

0x00400038 0x01097825 or $15,$8,$9 31 or $t7,$t0,$t1 # $t7<= 0x00f30023 or 0x005200e2 = 0x00f300e3

0x0040003c 0x0109c026 xor $24,$8,$9 32 xor $t8,$t0,$t1 # $t8<= 0x00f30023 xor 0x005200e2 = 0x00a100c1

0x00400040 0x0109c827 nor $25,$8,$9 33 nor $t9,$t0,$t1 # $t9<= 0x00f30023 nor 0x005200e2 = 0xff0cff1c

0x00400044 0x01090019 multu $8,$9 34 multu $t0,$t1 # Hi & Lo <= 0x00f30023 \* 0x005200e2 = 0x00004dd6e1bc1ee6

0x00400048 0x00008010 mfhi $16 35 mfhi $s0 # $s0<= 0x00004dd6

0x0040004c 0x00008812 mflo $17 36 mflo $s1 # $s1<= 0xe1bc1ee6

0x00400050 0x0109001b divu $8,$9 37 divu $t0,$t1 # Hi,Lo<= 0x00f30023 mod,/ 0x005200e2 = 0x4efe5f,0x00000002

0x00400054 0x250800ab addiu $8,$8,0x000000ab38 addiu $t0,$t0,0x00ab # $t0<= 0x00f30023 + 0x000000ab = 0x00f300ce

0x00400058 0x310800ab andi $8,$8,0x000000ab 39 andi $t0,$t0,0x00ab # $t0<= 0x00f300ce and 0x000000ab = 0x0000008a

0x0040005c 0x3908ffab xori $8,$8,0x0000ffab 40 xori $t0,$t0,0xffab # $t0<= 0x0000008a xor 0x0000ffab = 0x0000ff21

0x00400060 0x00084100 sll $8,$8,0x00000004 41 sll $t0,$t0,4 # $t0<= 0x000ff210 (deslocado 4 bits para a esquerda)

0x00400064 0x00084242 srl $8,$8,0x00000009 42 srl $t0,$t0,9 # $t0<= 0x000007f9 (deslocado 9 bits para a direita)

0x00400068 0x24120008 addiu $18,$0,0x000000043 addiu $s2,$zero,8 # $s2<= 0x00000008

0x0040006c 0x02594004 sllv $8,$25,$18 44 sllv $t0,$t9,$s2 # $t0<= 0x0007f900

0x00400070 0x02484004 sllv $8,$8,$18 45 sllv $t0,$t0,$s2 # $t0<= 0x07f90000

0x00400074 0x02484004 sllv $8,$8,$18 46 sllv $t0,$t0,$s2 # $t0<= 0xf9000000

0x00400078 0x00084103 sra $8,$8,0x00000004 47 sra $t0,$t0,4 # $t0<= 0xff900000

0x0040007c 0x02484007 srav $8,$8,$18 48 srav $t0,$t0,$s2 # $t0<= 0xffff9000

0x00400080 0x02484006 srlv $8,$8,$18 49 srlv $t0,$t0,$s2 # $t0<= 0x00ffff90

0x00400084 0x3c011001 lui $1,0x00001001 50 la $t0,array # coloca em $t0 o endere�o inicial do vetor array (0x10010000)

0x00400088 0x34280000 ori $8,$1,0x00000000

0x0040008c 0x91090006 lbu $9,0x00000006($8) 51 lbu $t1,6($t0) # $t1<= 0x000000ef (primero byte � terceiro byte do segundo elemento)

0x00400090 0x392900ff xori $9,$9,0x000000ff 52 xori $t1,$t1,0xff # $t1<= 0x00000010, inverte byte inferior

0x00400094 0xa1090006 sb $9,0x00000006($8) 53 sb $t1,6($t0) # segundo byte do segundo elemento do vetor <= 10 (resto n�o muda)

0x00400098 0x24080001 addiu $8,$0,0x0000000155 addiu $t0,$zero,0x1 # $t0<= 0x00000001

0x0040009c 0x00084023 subu $8,$0,$8 56 subu $t0,$zero,$t0 # $t0<= 0xffffffff

0x004000a0 0x0501000c bgez $8,0x0000000c 57 bgez $t0,loop # Esta instru��o nunca deve saltar, pois $t0 = -1

0x004000a4 0x0109582a slt $11,$8,$9 58 slt $t3,$t0,$t1 # $t3<= 0x00000001, pois -1 < 10

0x004000a8 0x0109582b sltu $11,$8,$9 59 sltu $t3,$t0,$t1 # $t3<= 0x00000000, pois (2^32)-1 > 10

0x004000ac 0x290b0001 slti $11,$8,0x0000000160 slti $t3,$t0,0x1 # $t3<= 0x00000001, pois -1 < 1

0x004000b0 0x2d0b0001 sltiu $11,$8,0x000000061 sltiu $t3,$t0,0x1 # $t3<= 0x00000000, pois (2^32)-1 > 1

0x004000b4 0x3c011001 lui $1,0x00001001 65 soma\_ct:la $t0,array # coloca em $t0 o endere�o do vetor (0x10010000)

0x004000b8 0x34280000 ori $8,$1,0x00000000

0x004000bc 0x3c011001 lui $1,0x00001001 66 la $t1,size # coloca em $t1 o endere�o do tamanho do vetor

0x004000c0 0x34290020 ori $9,$1,0x00000020

0x004000c4 0x8d290000 lw $9,0x00000000($9) 67 lw $t1,0($t1) # coloca em $t1 o tamanho do vetor

0x004000c8 0x3c011001 lui $1,0x00001001 68 la $t2,const # coloca em $t2 o endere�o da constante

0x004000cc 0x342a0024 ori $10,$1,0x00000024

0x004000d0 0x8d4a0000 lw $10,0x00000000($10)69 lw $t2,0($t2) # coloca em $t2 a constante

0x004000d4 0x19200006 blez $9,0x00000006 70 loop: blez $t1,end\_add # se/quando tamanho �/torna-se 0, fim do processamento

0x004000d8 0x8d0b0000 lw $11,0x00000000($8) 71 lw $t3,0($t0) # coloca em $t3 o pr�ximo elemento do vetor

0x004000dc 0x016a5821 addu $11,$11,$10 72 addu $t3,$t3,$t2 # soma constante

0x004000e0 0xad0b0000 sw $11,0x00000000($8) 73 sw $t3,0($t0) # atualiza no vetor o valor do elemento

0x004000e4 0x25080004 addiu $8,$8,0x0000000474 addiu $t0,$t0,4 # atualiza ponteiro do vetor. Lembrar, 1 palavra=4 posi��es de mem�ria

0x004000e8 0x2529ffff addiu $9,$9,0xffffffff75 addiu $t1,$t1,-1 # decrementa contador de tamanho do vetor

0x004000ec 0x08100035 j 0x004000d4 76 j loop # continua execu��o

0x004000f0 0x3c011001 lui $1,0x00001001 80 end\_add:li $sp,0x10010800 # Para poder simular o hardware, inicializa-se o $sp c/valor adequado

0x004000f4 0x343d0800 ori $29,$1,0x00000800

0x004000f8 0x27bdfffc addiu $29,$29,0xffffff81 addiu $sp,$sp,-4 # assume-se $sp inicializado, aloca espa�o na pilha

0x004000fc 0xafbf0000 sw $31,0x00000000($29)83 sw $ra,0($sp) # salva endere�o de retorno de quem chamou (trap handler)

0x00400100 0x0c100044 jal 0x00400110 84 jal sum\_tst # salta para subrotina sum\_tst

0x00400104 0x8fbf0000 lw $31,0x00000000($29)85 lw $ra,0($sp) # ao retornar, recupera endere�o de retorno da pilha

0x00400108 0x27bd0004 addiu $29,$29,0x00000086 addiu $sp,$sp,4 # atualiza apontador de pilha

0x0040010c 0x03e00008 jr $31 87 end: jr $ra # FIM DO PROGRAMA AQUI. volta para o "sistema operacional"

0x00400110 0x3c011001 lui $1,0x00001001 89 sum\_tst:la $t0,var\_a # pega endere�o da primeira vari�vel (pseudo-instru��o)

0x00400114 0x34280028 ori $8,$1,0x00000028

0x00400118 0x8d080000 lw $8,0x00000000($8) 90 lw $t0,0($t0) # toma o valor de var\_a e coloca em $t0

0x0040011c 0x3c011001 lui $1,0x00001001 91 la $t1,var\_b # pega endere�o da segunda vari�vel (pseudo-instru��o)

0x00400120 0x3429002c ori $9,$1,0x0000002c

0x00400124 0x8d290000 lw $9,0x00000000($9) 92 lw $t1,0($t1) # toma o valor de var\_b e coloca em $t1

0x00400128 0x01285021 addu $10,$9,$8 93 addu $t2,$t1,$t0 # soma var\_a com var\_b e coloca resultado em $t2

0x0040012c 0x27bdfff8 addiu $29,$29,0xffffff94 addiu $sp,$sp,-8 # aloca espa�o na pilha

0x00400130 0xafaa0000 sw $10,0x00000000($29)95 sw $t2,0($sp) # no topo da pilha coloca o resultado da soma

0x00400134 0xafbf0004 sw $31,0x00000004($29)96 sw $ra,4($sp) # abaixo do topo coloca o endere�o de retorno

0x00400138 0x3c010040 lui $1,0x00000040 97 la $t3,ver\_ev # pega endere�o da subrotina ver\_ev (pseudo-instru��o)

0x0040013c 0x342b0150 ori $11,$1,0x00000150

0x00400140 0x0160f809 jalr $31,$11 98 jalr $ra,$t3 # chama subrotina que verifica se resultado da soma � par

0x00400144 0x8fbf0004 lw $31,0x00000004($29)99 lw $ra,4($sp) # ao retornar, recupera endere�o de retorno da pilha

0x00400148 0x27bd0008 addiu $29,$29,0x000000100 addiu $sp,$sp,8 # atualiza apontador de pilha

0x0040014c 0x03e00008 jr $31 101 jr $ra # Rotina acaba AQUI. Retorna para quem chamou

0x00400150 0x8fab0000 lw $11,0x00000000($29)103 ver\_ev: lw $t3,0($sp) # tira dados to topo da pilha (par�metro)

0x00400154 0x316b0001 andi $11,$11,0x0000000104 andi $t3,$t3,1 # $t3 <= 1 se par�metro � �mpar, 0 caso contr�rio

0x00400158 0x03e00008 jr $31 105 jr $ra # e retorna

Data Segment

0x10010000 0xabcdef02 0xcd10ab17 0xefabcd34 0xbadcfeaa 0xdcfebacc 0xfebadc76 0xdefabc52 0xcbafed44

0x10010020 0x00000008 0xffffffff 0x000000ff 0x00000100 0x00000000 0x00000000 0x00000000 0x00000000