

```

1  # 1)
2  # a = 10;
3  # b = -1;
4  # a = a + 1;
5  # c = a + b;
6  #
7  # $s0 = a
8  # $s1 = b
9  # $s2 = c
10
11 .text
12 .globl main
13 main:
14 addi    $s0 , $zero , 10      # a = 10
15 addi    $s1 , $zero , -1      # b = -1
16 addi    $s0 , $s0 , 1         # a = a + 1
17 add $s2 , $s0 , $s1          # c = a + b
18
19 -----
20
21 # 2)
22 # x = 3;
23 # y = x * 4 ;
24 #
25 # $s0 = x
26 # $s1 = y
27
28 .text
29 .globl main
30 main:
31 addi    $s0 , $zero , 3       # x = 3
32 sll $s1 , $s0 , 2             # y = 4x
33
34 -----
35
36 # 3)
37 # x = 3;
38 # y = x * 1025;
39 #
40 # $s0 = x
41 # $s1 = y
42
43 .text
44 .globl main
45 main:
46 addi    $s0 , $zero , 3       # x = 3
47 sll $t0 , $s0 , 10            # $t0 = 4x
48 add $s1 , $t0 , $s0           # y = $t0 + x
49
50 -----
51
52 # 4)
53 # x = 3;
54 # y = x / 4;
55 #
56 # $s0 = x
57 # $s1 = y
58
59 .text
60 .globl main
61 main:
62 addi    $s0 , $zero , 3       # x = 3
63 srl $s1 , $s0 , 2             # $s1 = x / 4
64
65 -----
66
67 # 5)
68 # x = 305419896;
69 #
70 # $s0 = x
71
72 .text
73 .globl main

```

```

74  main:
75  ori $t0 , $zero , 0x1234    # $t0 = 0x1234
76  sll $t0 , $t0 , 16         # $t0 = $t0 << 16 = 0x12340000
77  ori $s0 , $t0 , 0x5678     # x = 0x12345678
78
79  -----
80
81  # 6)
82  # x = -1;
83  # y = x / 32;
84  #
85  # $s0 = x
86  # $s1 = y
87
88  .text
89  .globl main
90  main:
91  addi $t0 , $zero , 32      # t0 = 32
92  addi $s0 , $zero , -1     # x = -1
93  #sra $s1 , $s0 , 5        # y = x / 32
94  div $s0 , $t0             # $lo = x / 32
95  mflo $s1                 # y = x / 32
96
97  -----
98
99  # 7)
100 # A [ 12 ] = h + A [ 8 ];
101 #
102 # $s0 = h
103 # $s1 = &A[0]
104
105 .text
106 .globl main # rótulo onde o programa começa
107 j main
108
109 # Calcula o endereço de um elemento em um arranjo.
110 #
111 # @param Em $a0, Endereço de início do arranjo.
112 # @param Em $a1, Índice do elemento que se deseja buscar.
113 #
114 # @return Em $v0, o endereço do elemento no índice informado.
115
116 # calcular_endereco( endereco_base, indice )
117 calcular_endereco:
118 sll $t9 , $a1 , 2          # $t0 = indice * 4
119 add $v0 , $a0 , $t9        # $v0 = endereço + 4 * indice
120 jr $ra
121
122 main:
123 addi $s0 , $zero , 1       # h = 1
124 ori $s1 , $zero , 0x1001
125 sll $s1 , $s1 , 16
126 ori $s1 , $s1 , 0x0000     # A = 0x10010000
127
128 add $a0 , $zero , $s1      # Coloca o endereço base no primeiro argumento
129 addi $a1 , $zero , 8       # Coloca o índice no segundo argumento
130 jal calcular_endereco      # Calcula o endereço do elemento no arranjo
131 add $t9 , $zero , $v0      # Coloca o retorno da função em $t9
132
133 lw $t0 , 0 ($t9)           # Coloca A[8] em $t0
134 add $t0 , $s0 , $t0        # Coloca em $t0 h + A[8]
135
136 add $a0 , $zero , $s1      # Coloca o endereço base no primeiro argumento
137 addi $a1 , $zero , 12      # Coloca o índice no segundo argumento
138 jal calcular_endereco      # Calcula o endereço do elemento no arranjo
139 add $t9 , $zero , $v0      # Coloca o retorno da função em $t9
140
141 sw $t0 , 0 ($t9)           # Guarda h + A[8] em A[12]
142
143 .data
144 A0: .word 0
145 A1: .word 1
146 A2: .word 2

```

```

147 A3: .word 3
148 A4: .word 4
149 A5: .word 5
150 A6: .word 6
151 A7: .word 7
152 A8: .word 8
153 A9: .word 9
154 A10: .word 10
155 A11: .word 11
156 A12: .word 12
157
158 -----
159
160 # 8)
161 # h = k + A [ i ];
162 #
163 # $s0 = h
164 # $s1 = &A[0]
165 # $s2 = k
166 # $s3 = i
167
168 .text
169 .globl main # rótulo onde o programa começa
170 j main
171
172 # Calcula o endereço de um elemento em um arranjo.
173 #
174 # @param Em $a0, Endereço de início do arranjo.
175 # @param Em $a1, Índice do elemento que se deseja buscar.
176 #
177 # @return Em $v0, o endereço do elemento no índice informado.
178
179 # calcular_endereco( endereco_base, indice )
180 calcular_endereco:
181 sll $t9 , $a1 , 2 # $t0 = indice * 4
182 add $v0 , $a0 , $t9 # $v0 = endereço + 4 * indice
183 jr $ra
184
185 main:
186 # Inicialização das variáveis
187 addi $s0 , $zero , 1 # h = 1
188 addi $s2 , $zero , 1 # k = 1
189 addi $s3 , $zero , 1 # i = 1
190 ori $s1 , $zero , 0x1001
191 sll $s1 , $s1 , 16
192 ori $s1 , $s1 , 0x0000 # A = 0x10010000
193
194 add $a0 , $zero , $s1 # Coloca o endereço base no primeiro argumento
195 add $a1 , $zero , $s3 # Coloca o índice no segundo argumento
196 jal calcular_endereco # Calcula o endereço do elemento no arranjo
197 add $t9 , $zero , $v0 # Coloca o retorno da função em $t9
198
199 lw $t0 , 0 ($t9) # Coloca A[i] em $t0
200 add $s0 , $s2 , $t0 # h = k + A[i]
201
202 .data
203 A0: .word 0
204 A1: .word 1
205
206 -----
207
208 # 9)
209 # A [ j ] = h + A [ i ];
210 #
211 # $s0 = h
212 # $s1 = &A[0]
213 # $s2 = j
214 # $s3 = i
215
216 .text
217 .globl main # rótulo onde o programa começa
218 j main
219

```

```

220 # Calcula o endereco de um elemento em um arranjo.
221 #
222 # @param Em $a0, Endereço de início do arranjo.
223 # @param Em $a1, Índice do elemento que se deseja buscar.
224 #
225 # @return Em $v0, o endereço do elemento no índice informado.
226
227 # calcular_endereco( endereco_base, indice )
228 calcular_endereco:
229 sll $t9 , $a1 , 2      # $t0 = indice * 4
230 add $v0 , $a0 , $t9    # $v0 = endereco + 4 * indice
231 jr $ra
232
233 main:
234 addi $s0 , $zero , 1    # h = 1
235 ori $s1 , $zero , 0x1001
236 sll $s1 , $s1 , 16
237 ori $s1 , $s1 , 0x0000  # A = 0x10010000
238 addi $s2 , $zero , 1    # j = 1
239 addi $s3 , $zero , 1    # i = 1
240
241 add $a0 , $zero , $s1    # Coloca o endereço base no primeiro argumento
242 add $a1 , $zero , $s3    # Coloca o índice no segundo argumento
243 jal calcular_endereco    # Calcula o endereco do elemento no arranjo
244 add $t9 , $zero , $v0    # Coloca o retorno da função em $t9
245
246 lw $t0 , 0 ($t9)        # Coloca A[i] em $t0
247 add $t0 , $s0 , $t0     # Coloca em $t0 h + A[i]
248
249 add $a0 , $zero , $s1    # Coloca o endereço base no primeiro argumento
250 add $a1 , $zero , $s2    # Coloca o índice no segundo argumento
251 jal calcular_endereco    # Calcula o endereco do elemento no arranjo
252 add $t9 , $zero , $v0    # Coloca o retorno da função em $t9
253
254 sw $t0 , 0 ($t9)        # Guarda h + A[i] em A[j]
255
256 .data
257 A0: .word 0
258 A1: .word 1
259
260 -----
261
262 # 10)
263 # h = A [ i ] ;
264 # A[ i ] = A [ i + 1] ;
265 # A [ i + 1] = h ;
266 #
267 # $s0 = h
268 # $s1 = &A[0]
269 # $s3 = i
270
271 .text
272 .globl main # rótulo onde o programa começa
273 j main
274
275 # Calcula o endereco de um elemento em um arranjo.
276 #
277 # @param Em $a0, Endereço de início do arranjo.
278 # @param Em $a1, Índice do elemento que se deseja buscar.
279 #
280 # @return Em $v0, o endereço do elemento no índice informado.
281
282 # calcular_endereco( endereco_base, indice )
283 calcular_endereco:
284 sll $t9 , $a1 , 2      # $t0 = indice * 4
285 add $v0 , $a0 , $t9    # $v0 = endereco + 4 * indice
286 jr $ra
287
288 main:
289 addi $s0 , $zero , 1    # h = 1
290 ori $s1 , $zero , 0x1001
291 sll $s1 , $s1 , 16
292 ori $s1 , $s1 , 0x0000  # A = 0x10010000

```

```

293     addi      $s3 , $zero , 1      # i = 1
294
295     add $a0 , $zero , $s1          # Coloca o endereço base no primeiro argumento
296     add $a1 , $zero , $s3          # Coloca o índice no segundo argumento
297     jal calcular_endereco          # Calcula o endereço do elemento no arranjo
298     add $t9 , $zero , $v0          # Coloca o retorno da função em $t9
299
300     lw  $s0 , 0 ($t9)              # h = A[i]
301     lw  $t0 , 4 ($t9)              # $t0 = A[i + 1]
302     sw  $t0 , 0 ($t9)              # A[i] = $t0
303     sw  $s0 , 4 ($t9)              # A[i + 1] = h
304
305     .data
306     A0: .word 0
307     A1: .word 1
308     A2: .word 2
309
310     -----
311
312     # 11)
313     # j = 0;
314     # i = 10;
315     # do
316     # {
317     #     j = j + 1;
318     # }
319     # while ( j != i );
320     #
321     # $s2 = j
322     # $s3 = i
323
324     .text
325     .globl main # rótulo onde o programa começa
326
327     main:
328     # Inicialização das variáveis
329     addi      $s2 , $zero , 0      # j = 0
330     addi      $s3 , $zero , 10     # i = 10
331
332     do:
333     addi      $s2 , $s2 , 1        # j = j + 1
334     bne $s2 , $s3 , do            # goto do: if ($s2 != $s3)
335
336     -----
337
338     # 12)
339     # i = 2
340     # h = 100
341     # do
342     # {
343     #     a[i] = a[i - 1] + a[i - 2];
344     #     i = i + 1;
345     # } while ( i != h );
346     #
347     # $s0 = h
348     # $s1 = &A[0]
349     # $s3 = i
350
351     .text
352     .globl main # rótulo onde o programa começa
353     j main
354
355     # Calcula o endereço de um elemento em um arranjo.
356     #
357     # @param Em $a0, Endereço de início do arranjo.
358     # @param Em $a1, Índice do elemento que se deseja buscar.
359     #
360     # @return Em $v0, o endereço do elemento no índice informado.
361
362     # calcular_endereco( endereco_base, indice )
363     calcular_endereco:
364     sll $t9 , $a1 , 2              # $t0 = indice * 4
365     add $v0 , $a0 , $t9            # $v0 = endereço + 4 * indice

```

```

366 jr $ra
367
368 main:
369 addi    $s0 , $zero , 100      # h = 100
370 ori $s1 , $zero , 0x1001
371 sll $s1 , $s1 , 16
372 ori $s1 , $s1 , 0x0000      # A = 0x10010000
373 addi    $s3 , $zero , 2      # i = 2
374
375 do:
376
377 add $a0 , $zero , $s1      # Coloca o endereço base no primeiro argumento
378 add $a1 , $zero , $s3      # Coloca o índice no segundo argumento
379 jal calcular_endereco      # Calcula o endereço do elemento no arranjo
380 add $t9 , $zero , $v0      # Coloca o retorno da função em $t9
381
382 lw  $t0 , -4 ($t9)          # $t0 = A[i - 1]
383 lw  $t1 , -8 ($t9)          # $t1 = A[i - 2]
384 add $t0 , $t0 , $t1          # $t0 = A[i - 1] + A[i - 2]
385 sw  $t0 , 0 ($t9)          # A[i] = $t0
386 addi   $s3 , $s3 , 1      # i = i + 1
387
388 bne $s3 , $s0 , do          # goto do: if (i != h)
389
390 .data
391 A0: .word 1
392 A1: .word 1
393
394 -----
395
396 # 13)
397 # if ((h >= 50 && h <= 100) || (h >= 150 && h <= 200))
398 #     flag = 1;
399 # else
400 #     flag = 0;
401 #
402 # $s0 = h
403 # $s1 = flag
404
405 .text
406 .globl main # rótulo onde o programa começa
407
408 main:
409 addi    $s0 , $zero , 50      # h = 150
410
411 slti    $t1 , $s0 , 50        # set $t1 to 1 if h < 50
412 beq $t1 , 1 , flag0          # goto flag0 if $t1 = 1
413 slti    $t1 , $s0 , 100       # set $t1 to 1 if h < 100
414 beq $t1 , 1 , flag1          # goto flag1 if $t1 = 1
415 beq $s0 , 100 , flag1        # goto flag1 if h = 100
416
417 slti    $t1 , $s0 , 150       # set $t1 to 1 if h < 150
418 beq $t1 , 1 , flag0          # goto flag0 if $t1 = 1
419 slti    $t1 , $s0 , 200       # set $t1 to 1 if h < 200
420 beq $t1 , 1 , flag1          # goto flag1 if $t1 = 1
421 beq $s0 , 200 , flag1        # goto flag1 if h = 200
422
423 flag0:
424 addi    $s1 , $zero , 0      # flag = 0
425 j final
426
427 flag1:
428 addi    $s1 , $zero , 1      # flag = 1
429
430 final:
431
432 .data
433 A0: .word 0
434 A1: .word 2
435 A2: .word 1
436
437 -----
438

```

```

439 # 14)
440 # Algoritmo de ordenação seleção
441 #
442 # $s1 = &A[0]
443
444 .text
445 .globl main # rótulo onde o programa começa
446 j main
447
448 bubblesort: #($a0 = end. base, $a1 = inicio, $a2 = numero de elementos a serem
ordenados)
449 # $t9 = numero de elementos ordenados
450 # $t8 = endereco da bolha
451 # $t7 = indice final da ordenacao
452 # $t6 = indice da bolha
453 # $t5 = registrador para set on less than
454 # $t4 = elemento da bolha
455 # $t3 = elemento 'a esquerda da bolha (IMAGINANDO UM ARRANJO HORIZONTAL)
456 add $t9 , $zero , $zero # numero de elementos ordenados
457 add $t7 , $a1 , $a2 # indice final
458 addi $t7 , $t7 , -1 # indice final
459
460 while:
461 beq $t9 , $a2 , end # vai para end se ( numero de elementos ordenados ==
quantidade desejada (2° param) )
462 add $t6 , $zero , $t7 # indice bolha = indice final de ordenacao
463
464 for:
465 slt $t5 , $t9 , $t6 # $t5 = ( numero de elementos ordenados < indice
bolha ) ? 1 : 0
466 beq $t5 , $zero , endfor # vai para endfor se ( numero de elementos
ordenados >= bolha )
467 sll $t8 , $t6 , 2 # multiplica o indice por 4
468 addi $t6 , $t6 , -1 # indice bolha = indice bolha - 1
469 add $t8 , $a0 , $t8 # endereco bolha = endereco base + deslocamento
470 lw $t4 , 0 ($t8) # elemento da bolha = end. base[indice bolha]
471 lw $t3 , -4 ($t8) # elemento 'a esquerda da bolha = end.
base[indice bolha - 1]
472 slt $t5 , $t4 , $t3 # $t5 = ( elem_bolha < elem_a_esq_bolha ) ? 1 : 0
473 beq $t5 , $zero , for # vai para for se ( elem_bolha >=
elem_a_esq_bolha )
474 sw $t4 , -4 ($t8) # swap
475 sw $t3 , 0 ($t8) # swap
476
477 j for
478 endfor:
479
480 addi $t9 , $t9 , 1 # numero de elementos ordenados++
481 j while
482 end:
483 jr $ra
484
485 main:
486
487 addi $a0 , $zero , 0x1001
488 sll $a0 , $a0 , 16
489
490 lw $t0 , 0 ($a0) # carrega os valores da memoria
491 lw $t1 , 4 ($a0)
492 lw $t2 , 8 ($a0)
493
494 sw $t0 , 12 ($a0) # cria uma copia deles logo a frente
495 sw $t1 , 16 ($a0)
496 sw $t2 , 20 ($a0)
497
498 addi $a0 , $a0 , 12
499 addi $a1 , $zero , 0
500 addi $a2 , $zero , 3
501 jal bubblesort # ordena-os
502
503 lw $t1 , 4 ($a0) # pega o elemento do meio
504 sw $t1 , 12 ($a0) # guarda-o apos o vetor ordenado
505

```

```

506 .data
507 A0: .word 0
508 A1: .word 2
509 A2: .word 1
510
511 -----
512
513 # 15)
514 # $s0 = contador
515 # $s1 = end. base
516 # $s2 = copia end. base
517 # $s3 = soma
518 # $t0 = end. base[contador]
519
520 .text
521 .globl main
522
523 main:
524
525 addi $s1, $zero, 0x1001
526 sll $s1, $s1, 16 # end. base = 0x10010000
527 add $s2, $s1, $zero # copia end. base = end. base
528 addi $s0, $zero, 100 # contador = 100
529 add $s3, $zero, $zero # soma = 0
530
531 dowhile:
532 sw $s0, 0($s2) # *copia end. base = contador
533 lw $t0, 0($s2) # $t0 = *copia end. base
534 addi $s2, $s2, 4 # copia end. base = copia end. base + 4
535 addi $s0, $s0, -1 # contador = contador - 1
536 add $s3, $s3, $t0 # soma = soma + *copia end. base
537 bne $s0, $zero, dowhile # jump to do: if contador != 0
538
539 sw $s3, 0($s2) # *copia end. base = soma
540
541 -----

```