Programa 23

```
.data
A: .word -5   # Valor inicial de A

.text
main:
   lw $t0, A        # Carrega A para $t0
   jal abs_value    # Chama a função para obter o módulo
   sw $v0, A        # Armazena o módulo em A
   j end

abs_value:
   # Se $t0 < 0, muda o sinal; caso contrário, retorna o próprio valor
   bltz $t0, make_positive
   move $v0, $t0    # Caso positivo, retorna $t0
   jr $ra           # Retorna da função
make_positive:
   neg $v0, $t0     # Inverte o sinal
   jr $ra           # Retorna da função

end:
```

Programa 24

```
.data
Vetor: .word 1, 3, 5, 7, 9, 11, 13, 0, 2, 4, 6, 8, 10, 12
Soma: .word -1
Maior: .word -1

.text
main:
   la $t0, Vetor      # Carrega o endereço inicial de Vetor
   li $t1, 14         # Número de elementos no vetor
   li $t2, 0          # Inicializa a soma em 0
   lw $t3, ($t0)      # Carrega o primeiro elemento como maior inicial

loop:
   beq $t1, 0, end_loop   # Termina quando todos os elementos são processados
   lw $t4, ($t0)          # Carrega o próximo elemento do vetor
   add $t2, $t2, $t4      # Soma o elemento à soma total
   bgt $t4, $t3, new_max  # Verifica se é o novo maior
   j skip

new_max:
   move $t3, $t4          # Atualiza o maior elemento
```

```
skip:
   addi $t0, $t0, 4       # Avança para o próximo elemento
   subi $t1, $t1, 1       # Decrementa o contador
   j loop

end_loop:
   sw $t2, Soma           # Armazena a soma
   sw $t3, Maior          # Armazena o maior valor
```

Programa 25

```
.text
swap:
   sll $t1, $a1, 2       # Calcula o deslocamento k*4
   add $t2, $a0, $t1     # Calcula o endereço de v[k]
   lw $t0, 0($t2)        # Carrega v[k] em temp
   lw $t3, 4($t2)        # Carrega v[k+1]
   sw $t3, 0($t2)        # Armazena v[k+1] em v[k]
   sw $t0, 4($t2)        # Armazena temp em v[k+1]
   jr $ra                # Retorna da função
```

Programa 26

```
.text
maximo:
   lw $t0, 0($a0)       # Carrega o primeiro elemento como maior
   addi $a1, $a1, -1    # Decrementa o contador
   addi $a0, $a0, 4     # Avança para o próximo elemento

loop:
   beqz $a1, end_loop   # Se $a1 = 0, finaliza
   lw $t1, 0($a0)       # Carrega o próximo elemento
   blt $t1, $t0, skip   # Se não for maior, ignora
   move $t0, $t1        # Atualiza o maior

skip:
   addi $a0, $a0, 4     # Próximo elemento
   addi $a1, $a1, -1    # Decrementa o contador
   j loop

end_loop:
   move $v0, $t0        # Retorna o maior em $v0
   jr $ra
```

Programa 27

```
.data
x: .word 3
y: .word 0

.text
main:
   lw $t0, x          # Carrega x
   andi $t1, $t0, 1    # Verifica se x é ímpar
   beq $t1, 0, even    # Se par, vai para "even"

odd:
   mul $t2, $t0, $t0   # x^2
   mul $t3, $t2, $t0   # x^3
   mul $t4, $t3, $t0   # x^5
   addi $t4, $t4, 1    # x^5 + 1
   sub $t5, $t4, $t3   # x^5 - x^3 + 1
   sw $t5, y
   j end

even:
   mul $t2, $t0, $t0   # x^2
   mul $t3, $t2, $t0   # x^3
   mul $t4, $t3, $t0   # x^4
   sub $t5, $t4, $t2   # x^4 - 2x^2
   sw $t5, y

end:
```

Programa 28

```
.data
x: .word -2
y: .word 0

.text
main:
   lw $t0, x          # Carrega x
   blez $t0, non_pos   # Se x <= 0, vai para non_pos

positive:
   mul $t2, $t0, $t0   # x^2
   mul $t3, $t2, $t0   # x^3
   addi $t3, $t3, 1    # x^3 + 1
   sw $t3, y
   j end

non_pos:
```

```
    mul $t2, $t0, $t0   # x^2
    mul $t3, $t2, $t0   # x^3
    mul $t4, $t3, $t0   # x^4
    addi $t4, $t4, -1   # x^4 - 1
    sw $t4, y

end:
```