Rosenildo Pereira de Aguiar Furtado

Lista de Exercícios 03 – Estrutura de Repetição

Q1) Faça um programa que calcule e mostre os 10 primeiros números múltiplos de 3, considerando valores maiores que 0

```
.text

main:   addi $2, $0, 1

        addi $8, $0, 1

        addi $9, $0, 3

        addi $10, $0, 11

inifor: mul $4, $8, $9

        syscall

        addi $8, $8, 1

        addi $4, $0, ' '

        addi $2, $0, 11

        syscall

        addi $2, $0, 1

        beq $10, $8, fimfor

        j inifor


fimfor:  addi $2, $0, 10
```

```
        syscall
```

Q2) Faça um programa que solicite ao usuário um número para ser a referência e outro para ser a quantidade de valores a ser impresso no caso do programa da Q1. Para a mesma resposta, por exemplo, o usuário forneceria a referência 3 e a quantidade 10.

```
.text
main:   addi $2, $0, 5
        syscall
        addi $8, $2, 0
        addi $2, $0, 5
        syscall
        addi $9, $2, 0
        addi $10, $0, 1
        addi $11, $9, 1
inifor: addi $2, $0, 1
        mul $4, $8, $10
        syscall
        addi $10, $10, 1
        addi $4, $0, ' '
        addi $2, $0, 11
        syscall
        beq $10, $11, fimfor
        j inifor

fimfor:  addi $2, $0, 10
        syscall
```

Q3) Faça um programa que implementa um laço com um teste no início que conte de 0 a 10 imprimindo esses números, um em cada linha da saída.

```
.text
main:   addi $8, $0, 0
        addi $9, $0, 11

inifor: beq $8, $9, fimfor
        addi $2, $0, 1
```

```
        syscall
        addi $8, $8, 1
        addi $4, $0, '\n'
        addi $2, $0, 11
        syscall
        addi $4, $8, 0
        j inifor

fimfor: addi $2, $0, 10
        syscall
```

Q4) Faça um programa que leia do usuário um intervalo fechado e imprima os números pares desse intervalo (inclusive os limites).

```
.text
main:   addi $2, $0, 5
        syscall
        addi $8, $2, 0
        addi $2, $0, 5
        syscall
        addi $9, $2, 0

ifmenor: slt $25, $9, $8
        beq $25, $0, fimIfm
        addi $24, $8, 0
        addi $8, $9, 0
        addi $9, $24, 0
fimIfm: andi $10, $8, 1
        addi $11, $9, 1

if:     beq $10, $0, inifor
        addi $8, $8, 1

inifor: beq $8, $11, fimfor
```

```
        addi $4, $8, 0
        addi $2, $0, 1
        syscall
        addi $8, $8, 2
        addi $4, $0, ' '
        addi $2, $0, 11
        syscall

fimif:  j inifor

fimfor: addi $2, $0, 10
        syscall
```

Q5) Faça um programa que leia 10 valores e imprima a sua soma.

```
.text
main:   addi $8, $0, 0
        addi $9, $0, 0
        addi $10, $0, 10
inifor: beq $9, $10, fimFor
le:     addi $2, $0, 5
        syscall
        add $8, $8, $2
        addi $9, $9, 1
        j inifor

fimFor: addi $4, $8, 0
        addi $2, $0, 1
        syscall
        addi $2, $0, 10
        syscall
```

Q6) Faça um programa que leia números inteiros e calcule a soma. O laço do programa termina quando o usuário digita um valor negativo. Em seguida o programa imprime a soma dos valores digitados.

```
.text
main:   addi $8, $0, 0
        addi $9, $0, 0
        addi $10, $0, 10
while:  bne $9, $0, sai
le:     addi $2, $0, 5
        syscall
        add $8, $8, $2
        slt $9, $2, $0
        j while

sai:    addi $4, $8, 0
        addi $2, $0, 1
        syscall
        addi $2, $0, 10
        syscall
```

Q7) Faça um programa que leia números inteiros diferentes de zero e calcule a soma dos valores positivos. O laço do programa termina quando o usuário digita um valor zero. Em seguida o programa imprime a soma dos valores positivos digitados.

```
.text
main:   addi $2, $0, 5
        syscall
        addi $9, $0, 1
        addi $10, $0, 0

while:  beq $2, $0, sai
        slt $9, $2, $0
        bne $9, $0,le
        add $10, $10, $2
le:     addi $2, $0, 5
        syscall
        j while
```

```
sai:    addi $4, $10, 0
        addi $2, $0, 1
        syscall
        addi $2, $0, 10
        syscall
```

Q8) Faça um programa que leia números inteiros diferentes de zero e encontre o menor valor digitado e o maior valor digitado. O programa informa o maior e o menor, um em cada linha, quando o usuário digitar um 0.

```
.text
main:   addi $2, $0, 5
        syscall
        addi $8, $2, 0 # Maior
        addi $9, $2, 0 # menor

while:  beq $2, $0, sai
Tmenor:         slt $10, $2, $9
        beq $10, $0, Tmaior
        add $9, $0, $2
Tmaior: slt $10, $8, $2
        beq $10, $0, le
        add $8, $0, $2
le:     addi $2, $0, 5
        syscall
        j while

sai:    beq $8, $0, fim
        addi $4, $0, 'M'
        addi $2, $0, 11
        syscall
        addi $4, $8, 0
        addi $2, $0, 1
        syscall
```

```
        addi $4, $0, '\n'
        addi $2, $0, 11
        syscall
        addi $4, $0, 'm'
        addi $2, $0, 11
        syscall
        addi $4, $9, 0
        addi $2, $0, 1
        syscall
fim:    addi $2, $0, 10
        syscall
```

Q9) Faça um programa leia certa quantidade de números e imprima o maior deles e quantas vezes o maior número foi lido. A quantidade de números a serem lidos deve ser fornecido pelo usuário.

```
.text
main:   addi $2, $0, 5
        syscall
        addi $8, $2, 0 # Quantidade de números
        addi $9, $0, -999999999 # maior
        addi $10, $0, 0 # contador
        addi $11, $0 , 0 #numero de vezes que o maior aparece

while:  beq $10, $8, sai
le:     addi $2, $0, 5
        syscall
Tmenor:         slt $12, $2, $9
        bne $12, $0, cont
        bne $2, $9, maior
        addi $11, $11, 1
        j cont
maior:  add $9, $0, $2
        addi $11, $0, 1
cont:   addi $10, $10, 1
```

```
        j while

sai:    beq $8, $0, fim
        addi $4, $0, 'M'
        addi $2, $0, 11
        syscall
        addi $4, $9, 0
        addi $2, $0, 1
        syscall
        addi $4, $0, '\n'
        addi $2, $0, 11
        syscall
        addi $4, $0, 'q'
        addi $2, $0, 11
        syscall
        addi $4, $11, 0
        addi $2, $0, 1
        syscall
fim:    addi $2, $0, 10
        syscall
```

Q10) Faça um programa para calcular o MDC de dois números fornecidos pelo usuário, usando o algoritmo de Euclides (busque na Internet o funcionamento desse algoritmo).

```
.text   # Calculo do MDC pelo algoritmo de Euclides, tirado de:
        #https://pt.wikipedia.org/wiki/Algoritmo_de_Euclides#:~:text=O%20MDC%20de%20dois%20números,número%20for%20subtraído%20ao%20maior.&text=Nesse%20momento%2C%20o%20MDC%20é,número%20inteiro%2C%20maior%20que%20zero.
main:   addi $2, $0, 5
        syscall
        addi $8, $2, 0 # a
        addi $2, $0, 5
        syscall
        addi $9, $2, 0 # b
```

```
while:  beq $9, $0, sai
        div $8, $9 # r = a % b
        addi $8, $9, 0 # a = b
        mfhi $9 # b = r
        j while

sai:    addi $4, $8, 0
        addi $2, $0, 1
        syscall
fim:    addi $2, $0, 10
        syscall
```

Q11) Faça um programa que encontre o primeiro múltiplo de 11, 13 ou 17 após um número fornecido pelo usuário.

```
.text
main:   addi $2, $0, 5
        syscall
        addi $8, $2, 0 # n
        addi $9, $0, 11 # constante 11
        addi $10, $0, 13 # constante 13
        addi $11, $0, 17 # constante 17
        addi $8, $8, 1

while:  beq $9, $0, sai
        div $8, $9
        mfhi $12
        beq $12, $0, sai
        div $8, $10
        mfhi $12
        beq $12, $0, sai
        div $8, $11
        mfhi $12
        beq $12, $0, sai
```

```
        addi $8, $8, 1
        j while

sai:    addi $4, $8, 0
        addi $2, $0, 1
        syscall
fim:    addi $2, $0, 10
        syscall
```

Q12) Faça um programa que imprima os n primeiros números da série de Fibonacci, sendo n fornecido pelo usuário.

```
.text
main:   addi $2, $0, 5
        syscall
        addi $8, $2, 0 # n
        addi $9, $0, 0 # fb1
        addi $10, $0, 1 # fb2
        addi $11, $0, 0 # soma
        addi $12, $0, 0 # contador

        beq $8, $12, fim
        add $4, $9, $0
        addi $2, $0, 1
        syscall
        addi $4, $0, ' '
        addi $2, $0, 11
        syscall
        addi $12, $12, 1 # contador

        beq $8, $12, fim
        add $4, $10, $0
        addi $2, $0, 1
        syscall
```

```
        addi $4, $0, ' '
        addi $2, $0, 11
        syscall
        addi $12, $12, 1 # contador

while:  beq $8, $12, fim
        add $4, $9, $10
        add $9, $0, $10
        add $10, $4, $0

        addi $2, $0, 1
        syscall
        addi $4, $0, ' '
        addi $2, $0, 11
        syscall
        addi $12, $12, 1 # contador
        j while

fim:    addi $2, $0, 10
        syscall
```

Q13) Faça um programa para calcular o fatorial de um número

```
.text
main:   addi $2, $0, 5
        syscall
        addi $8, $2, 0 # n
        addi $9, $0, 1 # fatorial

while:  beq $8, $0, sai
        mul $9, $9, $8
        subi $8, $8, 1
        j while
```

```
sai:    add $4, $9, $0
        addi $2, $0, 1
        syscall

fim:    addi $2, $0, 10
        syscall
```

Q14) Faça um programa que calcule o(s) terno(s) pitagórico(s) a, b, c, para o qual a+b+c é um valor fornecido pelo usuário. Um terno pitagórico é uma tríade tal que $a^2+b^2=c^2$. Por exemplo, o usuário forneceu o valor 12, ora, um terno pitagórico para esse caso é a=3, b=4 e c=5, pois $3^2+4^2=5^2$ e 3+4+5 =12.

```
.text
main:   addi $2, $0, 5
        syscall
        add $8, $0, $2 # n
        addi $9, $0, 3 # a
        addi $10, $0, 4 # b
        sub $11, $8, $9
c:      sub $11, $11, $10
        addi $12, $0, 0 # a + b + c
        addi $13, $0, 0 # a2
        addi $14, $0, 0 # b2
        addi $15, $0, 0 # c2
        addi $16, $0, 0 # a2 + b2
        srl $17, $8, 1 # n/2
        addi $17, $17, -1 # n/2-1
        addi $18, $0, 3 # constante 3
        div $8, $18
        mflo $18 # n/3

forA:   slt $24, $9, $10
        beq $24, $0, naoTem
a2:     mul $13, $9, $9
        addi $10, $9, 1
```

```
forB:    slt $25, $10, $11
         beq $25, $0, fimForB
b2:      mul $14, $10, $10
         sub $11, $8, $9
         sub $11, $11, $10
c2:      mul $15, $11, $11
         add $16, $13, $14
         beq $16, $15, sai

         addi $10, $10, 1
         j forB

fimForB: addi $9, $9, 1
         j forA

naoTem: addi $4, $0, 'N'
         addi $2, $0, 11
         syscall
         addi $4, $0, 'a'
         syscall
         addi $4, $0, 'o'
         syscall
         addi $4, $0, ' '
         syscall
         addi $4, $0, 't'
         syscall
         addi $4, $0, 'e'
         syscall
         addi $4, $0, 'm'
         syscall
         j fim
```

```
sai:    addi $4, $0, 'a'
        addi $2, $0, 11
        syscall
        addi $4, $9, 0
        addi $2, $0, 1
        syscall
        addi $4, $0, ' '
        addi $2, $0, 11
        syscall
        addi $4, $0, 'b'
        addi $2, $0, 11
        syscall
        addi $4, $10, 0
        addi $2, $0, 1
        syscall
        addi $4, $0, ' '
        addi $2, $0, 11
        syscall
        addi $4, $0, 'c'
        addi $2, $0, 11
        syscall
        addi $4, $11, 0
        addi $2, $0, 1
        syscall
        j fim

fim:    addi $2, $0, 10
        syscall
```

Q15) *Escreva um programa que leia um número inteiro positivo n e em seguida imprima n linhas do chamado Triangulo de Floyd. Por exemplo, para n = 6, temos:

```
1
2 3
4 5 6
```

```
7 8 9 10
11 12 13 14 15
16 17 18 19 20 21
```

```
.text
main:   addi $2, $0, 5
        syscall
        add $8, $0, $2 # n
        addi $9, $0, 1 # contador 1
        addi $10, $0, 0 # contador linhas
        addi $11, $0, 0 # contador colunas

        slt $12, $8, $0
        bne $12, $0, sai
        addi $25, $10, 1

forA:   beq $8, $10, sai
        addi $13, $10, 1
forB:   beq $25, $11, fimForB
        addi $4, $9, 0
        addi $2, $0, 1
        syscall
        addi $4, $0, ' '
        addi $2, $0, 11
        syscall

        addi $9, $9, 1
        addi $11, $11, 1
        j forB

fimForB: addi $10, $10, 1
        addi $25, $10, 1
        addi $11, $0, 0 # contador colunas
        addi $4, $0, '\n'
        addi $2, $0, 11
        syscall
        j forA
sai:
fim:    addi $2, $0, 10
        syscall
```

Q16) *Faça um programa que conte e imprima quantos números primos existem entre a e b, onde a e b são números informados pelo usuário.

```
.text
main:   addi $2, $0, 5
```

```
        syscall
        add $8, $0, $2 # a
        addi $2, $0, 5
        syscall
        add $9, $0, $2 # b
        addi $10, $0, 2 # constante 2

        slt $11, $9, $8 # Testa se b<a
        beq $11, $0, ok
        addi $25, $8, 0
        addi $8, $9, 0
        addi $9, $25, 0
ok:     slt $11, $8, $10 # Testa se a<2 e inicia o contador
        add $8, $8, $11
        addi $12, $0, 3 # constate 3
        not $13, $8
        andi $14, $13, 1
        add $8, $8, $14

forA:   slt $25, $8, $9
        beq $25, $0, sai # enquanto a<b
        srl $15, $8, 1
forB:   slt $24, $12, $15
        beq $24, $0, fimForB
        div $8, $12
        mfhi $16
        beq $16, $0, prox
        addi $12, $12, 2
        j forB

fimForB: addi $11, $11, 1
prox:   addi $12, $0, 3 # constate 3
        addi $8, $8, 2

        j forA
sai:    addi $4, $0, 'q'
        addi $2, $0, 11
        syscall
        addi $4, $0, 'p'
        addi $2, $0, 11
        syscall
        addi $4, $0, '='
        addi $2, $0, 11
        syscall
        addi $4, $11, 0
```

```
        addi $2, $0, 1
        syscall
fim:    addi $2, $0, 10
        syscall
```

Q17) * Faça um programa que calcule o maior número palíndromo feito a partir do produto de dois números de 3 algarismos. Ex.: O maior palíndromo feito a partir do produto de dois números de dois algarismos é 9009 = 91*99.

```
.text
main:   addi $2, $0, 5
        syscall
        add $8, $0, $2 # a
        addi $2, $0, 5
        syscall
        add $9, $0, $2 # b
        mul $10, $8, $9 # a * b
        addi $11, $0, 100000
        addi $12, $0, 10
        addi $13, $0, 6

forA:   beq $13, $0, continue
        div $10, $11
        mflo $14
        mfhi $25
        bne $14, $0, continue #verifica quantos algarismos tem a * b
        div $11, $12
        mflo $11
        addi $13, $13, -1
        j forA

continue: addi $24, $0, 1
        beq $13, $24, sai
        beq $25, $0, reduz
        andi $15, $13, 1
        srl $16, $13, 1

while:  beq $10, $0, sai
        addi $17, $10, 0
        addi $18, $11, 0
        addi $19, $0, 0
compara: beq $19, $16, sai
        div $17, $18
        mfhi $17
        mflo $20
```

```
        div $17, $12
        mflo $17
        mfhi $21
        bne $20, $21, prox
        addi $22, $0, 100
        div $18, $22
        mflo $18
        addi $19, $19, 1
        j compara
prox:   addi $10, $10, -1
        j while
reduz:  addi $10, $10, -1
sai:    addi $4, $10, 0
        addi $2, $0, 1
        syscall
fim:    addi $2, $0, 10
        syscall
```

Q18) *Faça um programa que receba um ano e informe a data da Páscoa, a data do Carnaval e a data de Corpus Christi desse ano.

```
.text # Algoritmo de Meeus/Jones/Butcher (Wikipedia)
main:   addi $2, $0, 5
        syscall
        addi $8, $0, 100 # constante 100
        addi $9, $0, 31 # constante 31
        addi $10, $0, 25 # constante 25
        addi $11, $0, 114 # constante 114
        addi $12, $0, 8 # constante 8
        addi $13, $0, 19 # constante 19

        div $2, $13
a:      mfhi $14 # a = ANO MOD 19
        div $2, $8
b:      mflo $15 # b = ANO \ 100
c:      mfhi $16 # c = ANO MOD 100
        addi $17, $0, 4 # constante 4
        div $15, $17
d:      mflo $18 # d = b \ 4
e:      mfhi $19 # e = b MOD 4
        add $20, $15, $12
        div $20, $10
f:      mflo $21 # f = (b + 8) \ 25
        addi $17, $0, 3 # constante 3
        sub $22, $15, $21
```

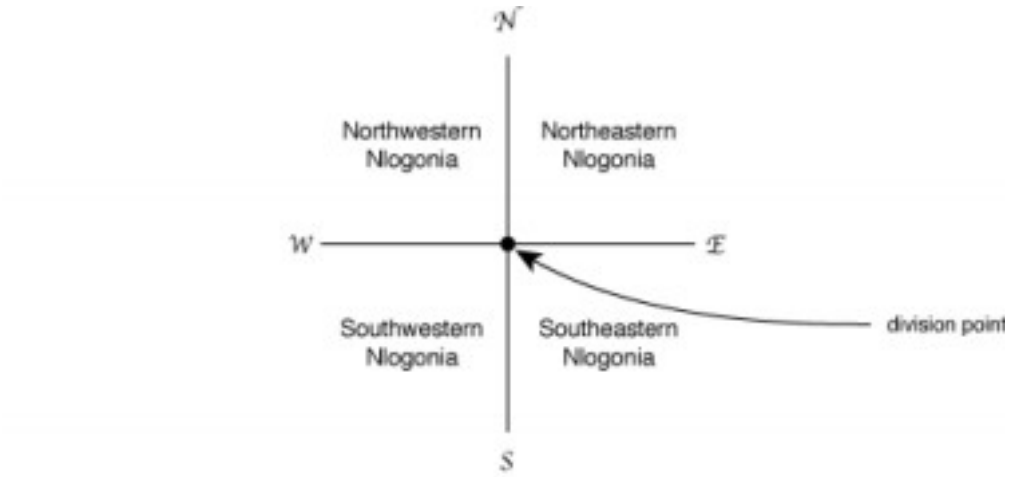
```
        addi $22, $22, 1
        div $22, $17
g:      mflo $23 # g = (b - f + 1) \ 3
        mult $13, $14
        mflo $24
        add $24, $24, $15
        sub $24, $24, $18
        sub $24, $24, $23
        addi $24, $24, 15
        addi $17, $0, 30 # constante 30
        div $24, $17
h:      mfhi $24 # h = (19 × a + b - d - g + 15) MOD 30
        addi $17, $0, 4 # constante 4
        div $16, $17
i:      mflo $25 # i = c \ 4
k:      mfhi $22 # k = c MOD 4
        add $10, $25, $19
        sll $10, $10, 1
        addi $10, $10, 32
        sub $10, $10, $24
        sub $10, $10, $22
        addi $17, $0, 7 # constante 7
        div $10, $17
L:      mfhi $10 # L = (32 + 2 × e + 2 × i - h - k) MOD 7
        addi $20, $0, 11 # constante 11
        mult $24, $20
        mflo $8
        addi $20, $0, 22 # constante 22
        mult $10, $20
        mflo $9
        add $8, $8, $9
        add $8, $8, $14
        addi $20, $0, 451 # constante 451
        div $8, $20
m:      mflo $8 # m = (a + 11 × h + 22 × L) \ 451
        mult $8, $17
        mflo $9
        add $11, $10, $24
        sub $11, $11, $9
        addi $11, $11, 114
        addi $20, $0, 31 # constante 31
        div $11, $20
MES:    mflo $11 # MÊS = (h + L - 7 × m + 114) \ 31
        mfhi $12
DIA:    addi $12, $12, 1 # DIA = 1+ (h + L - 7 × m + 114)MOD 31
```

```
ANO:    addi $25, $2, 0 # guarda o ano digitado

saida:  addi $4, $0, 'P'
        addi $2, $0, 11
        syscall
        addi $4, $0, '='
        syscall
        addi $4, $12, 0
        addi $2, $0, 1
        syscall
        addi $4, $0, '/'
        addi $2, $0, 11
        syscall
        addi $4, $11, 0
        addi $2, $0, 1
        syscall
        addi $4, $0, '/'
        addi $2, $0, 11
        syscall
        addi $4, $25, 0
        addi $2, $0, 1
        syscall
        addi $4, $0, '\n'
        addi $2, $0, 11
        syscall
        addi $4, $0, 'C'
        syscall
        addi $4, $0, '='
        syscall

        addi $23, $12, 0
        addi $24, $11, 0
        addi $8, $25, 0


bissex: addi $10, $0, 4 # constante 4
        addi $11, $0, 100 # constante 100
        addi $12, $0, 400 # constante 400
        addi $22, $0, 47 # constante N
        div $8, $10
        mfhi $9

se:     bne $9, $0, fim # não é divisivel por 4?
        addi $22, $0, 46
        div $8, $11
```

```
        mfhi $9
senaoSe: bne $9, $0, fim # não é divisivel por 100?
        div $8, $12
        mfhi $9
senao:  beq $9, $0, fim # é divisivel por 400?
        addi $22, $0, 47

fim:    sub $22, $22, $23
        subi $21, $24, 1
        addi $19, $0, 31
        slt $20, $22, $19
        sub $21, $21, $20
        bne $20, $0, carnav
        sub $22, $22, $19
        addi $18, $0, 28
        sub $22, $18, $22
carnav: addi $4, $22, 0
        addi $2, $0, 1
        syscall
        addi $4, $0, '/'
        addi $2, $0, 11
        syscall
        addi $4, $21, 0
        addi $2, $0, 1
        syscall
        addi $4, $0, '/'
        addi $2, $0, 11
        syscall
        addi $4, $25, 0
        addi $2, $0, 1
        syscall
        addi $4, $0, '\n'
        addi $2, $0, 11
        syscall
        addi $4, $0, 'C'
        syscall
        addi $4, $0, 'C'
        syscall
        addi $4, $0, '='
        syscall

        addi $24, $24, 1
        addi $17, $0, 30
        sub $23, $17, $23
        sll $17, $17, 1
```

```
        sub $23, $17, $23
        slt $16, $19, $23
        beq $16, $0, corpcr
        sub $23, $23, $19
        addi $24, $24, 1

corpcr: addi $4, $23, 0
        addi $2, $0, 1
        syscall
        addi $4, $0, '/'
        addi $2, $0, 11
        syscall
        addi $4, $24, 0
        addi $2, $0, 1
        syscall
        addi $4, $0, '/'
        addi $2, $0, 11
        syscall
        addi $4, $25, 0
        addi $2, $0, 1
        syscall

        addi $2, $0, 10
        syscall
```

Q19) *Depois de séculos de escaramuças entre os quatro povos habitantes da Nlogônia, e de dezenas de anos de negociações envolvendo diplomatas, políticos e as forças armadas de todas as partes interessadas, com a intermediação da ONU, OTAN, G7 e SBC, foi finalmente decidida e aceita por todos a maneira de dividir o país em quatro territórios independentes.
Ficou decidido que um ponto, denominado ponto divisor, cujas coordenadas foram estabelecidas nas negociações, definiria a divisão do país, da seguinte maneira. Duas linhas, ambas contendo o ponto divisor, uma na direção norte-sul e uma na direção leste-oeste, seriam traçadas no mapa, dividindo o país em quatro novos países. Iniciando no quadrante mais ao norte e mais ao oeste, em sentido horário, os novos países seriam chamados de Nlogônia do Noroeste, Nlogônia do Nordeste, Nlogônia do sudeste e Nlogônia do Sudoeste.



A ONU determinou que fosse disponibilizada uma página na Internet para que os habitantes pudessem consultar em qual dos novos países suas residências estão, e você foi contratado para ajudar a implementar o sistema.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém um inteiro *K* indicando o número de consultas que serão realizadas (*0 < K ≤ 103*). A segunda linha de um caso de teste contém dois números inteiros *N* e *M* representando as coordenadas do ponto divisor (*-104 < N, M < 104*). Cada uma das K linhas seguintes contém dois inteiros *X* e *Y* representando as coordenadas de uma residência (*-104 ≤ X, Y ≤ 104*). Em todas as coordenadas dadas, o primeiro valor corresponde à direção leste-oeste, e o segundo valor corresponde à direção norte-sul.

O final da entrada é indicado por uma linha que contém apenas o número zero.

Saída

Para cada caso de teste da entrada seu programa deve imprimir uma linha contendo a palavra:

- **DV se a residência encontra-se em cima de uma das linhas divisórias (norte-sul ou leste oeste);**
- **NO se a residência encontra-se na Nlogônia do Noroeste;**
- **NE se a residência encontra-se na Nlogônia do Nordeste;**
- **SE se a residência encontra-se na Nlogônia do Sudeste;**
- **SO se a residência encontra-se na Nlogônia do Sudoeste.**

Exemplo de Entrada Exemplo de Saída

```
3
2 1
10 10
-10 1
0 33
4
-1000 -1000
-1000 -1000
0 0
```

```
-2000 -10000
-999 -1001
0
```

Extraído da Maratona de Programação da SBC 2008.

E DV C DV E SO SE

Adapte esse problema e o próximo para ler um valor a cada linha, para facilitar o código em assembly. Por exemplo, no primeiro conjunto do exemplo temos: 3

```
2 1
10 10
-10 1
0 33
```

Devemos ler da seguinte forma:

```
3
2
1
10
10
-10
1
0
33
```

Ao término da digitação de cada ponto já se espera a impressão da saída. Nesse caso, após os dois valores 10 já teremos a saída NE. Após o 1 que vem depois de -10, já se espera a impressão DV e assim sucessivamente.

```
.text
main:   addi $2, $0, 5
        syscall
        add $8, $0, $2 # k
        addi $2, $0, 5
        syscall
        add $9, $0, $2 # px
        addi $2, $0, 5
        syscall
        add $10, $0, $2 # py
        addi $11, $0, -104 # constante -104
        addi $12, $0, 104 # constante 104
        addi $13, $0, 0 # contador
```

```
while:  beq $13, $8, fim
        addi $2, $0, 5
        syscall
        add $14, $0, $2 # px
        addi $2, $0, 5
        syscall
        add $15, $0, $2 # px
        beq $14, $9, DV
        beq $15, $10, DV
        slt $25, $14, $9
        beq $25, $0, testey
        slt $24, $15, $10
        beq $24, $0, NO
        j SO

testey: slt $24, $15, $10
        beq $24, $0, NE
        j SE

DV:     addi $4, $0, 'D'
        addi $2, $0, 11
        syscall
        addi $4, $0, 'V'
        syscall
        addi $4, $0, ' '
        syscall
        addi $13, $13, 1
        j while

NO:     addi $4, $0, 'N'
        addi $2, $0, 11
        syscall
        addi $4, $0, 'O'
        syscall
        addi $4, $0, ' '
        syscall
        addi $13, $13, 1
        j while

NE:     addi $4, $0, 'N'
        addi $2, $0, 11
        syscall
```

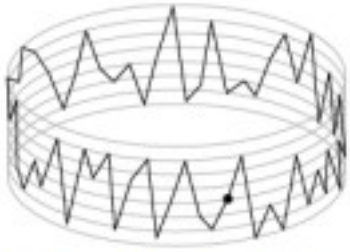
```
        addi $4, $0, 'E'
        syscall
        addi $4, $0, ' '
        syscall
        addi $13, $13, 1
        j while

SO:     addi $4, $0, 'S'
        addi $2, $0, 11
        syscall
        addi $4, $0, 'O'
        syscall
        addi $4, $0, ' '
        syscall
        addi $13, $13, 1
        j while

SE:     addi $4, $0, 'S'
        addi $2, $0, 11
        syscall
        addi $4, $0, 'E'
        syscall
        addi $4, $0, ' '
        syscall
        addi $13, $13, 1
        j while
sai:
fim:    addi $2, $0, 10
        syscall
```

Q20) *Um *loop musica*l é um trecho de música que foi composto para repetir continuamente (ou seja, o trecho inicia novamente toda vez que chega ao final), sem que se note descontinuidade. Loops são muito usados na sonorização de jogos, especialmente jogos casuais pela internet.

Loops podem ser digitalizados por exemplo utilizando PCM. PCM, do inglês Pulse Code Modulation, é uma técnica para representação de sinais analógicos, muito utilizada em áudio digital. Nessa técnica, a magnitude do sinal é amostrada a intervalos regulares de tempo, e os valores amostrados são armazenados em sequência. Para reproduzir a forma de onda amostrada, o processo é invertido (demodulação). Fernandinha trabalha para uma empresa que desenvolve jogos e compôs um bonito loopmusical, codificando-o em PCM. Analisando a forma de onda do seu loop em um software de edição de áudio, Fernandinha ficou curiosa ao notar a quantidade de "picos" existentes. Um pico em uma forma de onda é um valor de uma amostra que representa um máximo ou mínimo local, ou seja, um ponto de inflexão da

forma de onda. A figura abaixo ilustra (a) um exemplo de forma de onda e (b) o loop formado com essa forma de onda, contendo 48 picos.

(a) Uma forma de onda



(b) A mesma forma de onda como um loop

Fernandinha é uma amiga muito querida e pediu sua ajuda para determinar quantos picos existem no seu loop musical.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém um inteiro ***N***, representando o número de amostras no loop musical de Fernandinha ($2 \leq N \leq 104$). A segunda linha contém ***N*** inteiros *Hi*, separados por espaços, representando a sequência de magnitudes das amostras ($-104 \leq Hi \leq 104$ para $1 \leq i \leq N$, $H1 \neq HN$ e $Hi \neq Hi+1$ para $1 \leq i < N$). Note que *H1* segue *HN* quando o loop é reproduzido.

O final da entrada é indicado por uma linha que contém apenas o número zero.

Saída

Para cada caso de teste da entrada seu programa deve imprimir uma única linha, contendo apenas um inteiro, o número de picos existentes no loop musical de Fernandinha.

Exemplo de Entrada Exemplo de Saída

```
2
1 -3
```

```
2
```

```
2
```

6

4

40 0 -41 0 41 42

4

300 450 449 450

0

Maratona de Programação da SBC 2008

Obs.: como no exercício anterior, as entradas são feitas uma a uma.

```
6
40 0 -41 0 41 42
```

Serão entradas

```
6
40
0
-41
0
41
42
```

Ao fim das quais já se produz a saída correspondente. 2