Rosenildo Pereira de Aguiar Furtado

Lista de Exercícios 02 – Estrutura de Decisão

Q1) Faça um programa para ler dois números e informe qual deles é o maior.

```
.test

main:   addi $2, $0, 5

        syscall

        add $8, $2, $0

        addi $2, $0, 5

        syscall

        add $4, $2, $0

        slt $10, $4, $8

        beq $10, $0, fim

        add $4, $8, $0

fim:    addi $2, $0, 1

        syscall



        addi $2, $0, 10

        syscall
```

Q2) Faça um programa para ler um número inteiro. Se o número for positivo, imprima o dobro do número, se for negativo, imprima o quadrado do número.

```
.test
```

```
main:   addi $2, $0, 5
        syscall
        add $8, $2, $0
        beq $8, $0, fim
        slt $10, $8, $0
        sll $4, $8, 1
        beq $10, $0, res
        mul $4, $8, $8
res:    addi $2, $0, 1
        syscall

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

Q3) Escreva um programa para ler dois números inteiros e mostrar na tela o maior deles, bem como a diferença entre eles (a diferença é sempre um valor positivo)

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

        sub $11, $9, $8
        slt $10, $9, $8
        add $4, $9, $0
```

```
        beq $10, $0, maior
        add $4, $8, $0
        sub $11, $8, $9
maior:  addi $2, $0, 1
        syscall
        addi $4, $0, '\n'
        addi $2, $0, 11
        syscall
        add $4, $11, $0
        addi $2, $0, 1
        syscall
        addi $2, $0, 10
        syscall
```

Q4) Faça um programa que leia dois números e escreva a relação de grandeza entre eles. Ex. 345 e 23 gera a saída 345>23. Ex.: 24 e 38 gera a saída 24<38. Ex.: 12 e 12 gera a saída 12=12

```
.test
main:   addi $2, $0, 5
        syscall
        add $8, $2, $0
        addi $2, $0, 5
        syscall
        add $9, $2, $0
        addi $10, $0, 61 # Constante 61 equivale a '=' em ascII

        slt $11, $9, $8 # ver se n2<n1
```

```
        slt $12, $8, $9 # ver se n1<n2

        sub $13, $11, $12 # Va ser usado na expressão c = 61 - $11 + $10, onde c é o caracter que será impresso



        add $4, $8, $0

        addi $2, $0, 1

        syscall



        add $4, $10, $0

        beq $8, $9, fim # testa se os números são iguais

        add $4, $10, $13 # atualiza caracter

fim:    addi $2, $0, 11

        syscall



        add $4, $9, $0

        addi $2, $0, 1

        syscall

        addi $2, $0, 10

        syscall
```

Q5) Faça um programa que receba três notas (entre 0 e 100) e calcule a média ponderada dessas notas com pesos 1, 2 e 3. Informe a média e se o aluno foi aprovado, escreva após a média o a letra A. Caso o aluno seja reprovado, informe, após a média, a letra R. A média para aprovação é 60.

```
.test

main:   addi $2, $0, 5

        syscall

        add $8, $2, $0
```

```
        addi $2, $0, 5
        syscall
        add $9, $2, $0
        addi $2, $0, 5
        syscall
        add $10, $2, $0


        addi $11, $0, 3 # Peso 3
        addi $12, $0, 'A' # caracter A
        sll $13, $9, 1 # multiplicou a segunda nota por 2
        mul $14, $10, $11 # multiplicou a terceira nota por 3
        add $15, $8, $13
        add $15, $15, $14 # somou $8 + 2*$9 + 3*$10
        addi $11, $0, 6 # constante 6
        div $15, $11
media:  mflo $4 # media
        addi $2, $0, 1
        syscall
        addi $11, $0, 60 # constante 60
        slt $16, $4, $11 # ver se media<60
        beq $16, $0, fim
        addi $12, $0, 'R' # atualiza caracter
fim:    add $4, $12, $0
        addi $2, $0, 11
```

```
        syscall

        addi $2, $0, 10

        syscall
```

Q6) Faça um programa que leia a idade (em anos) e o tempo de serviço de um trabalhador. Informe se ele pode se aposentar (imprima S se sim, e N se não). As condições para aposentadoria são: 1) ter, ao menos, 65 anos; OU 2) ter trabalhado 40 anos; OU ter pelo menos 60 anos e mais de 35 anos.

```
.test

main:   addi $2, $0, 5

        syscall

        add $8, $2, $0

        addi $2, $0, 5

        syscall

        add $9, $2, $0

        addi $2, $0, 5


        addi $10, $0, 60 # constante 60

        addi $11, $0, 65 # constante 65

        addi $12, $0, 35 # constante 65

        addi $13, $0, 40 # constante 65

        addi $4, $0, 'S' # constante S


se:     slt $15, $8, $11 # ver se ano<65

        beq $15, $0, fim

senaoSe: slt $16, $9, $13 # ver se trabalho<40

        beq $16, $0, fim
```

```
senao:  slt $17, $8, $10 # ver se ano<60
        slt $18, $9, $12 # ver se trabalho<35
        add $19, $17, $18
        beq $19, $0, fim
        addi $4, $0, 'N' # constante N
fim:    addi $2, $0, 11
        syscall
        addi $2, $0, 10
        syscall
```

Q7) Faça um programa que leia um ano e informe se esse ano é bissexto.

```
.test
main:   addi $2, $0, 5
        syscall
        add $8, $2, $0

        addi $10, $0, 4 # constante 4
        addi $11, $0, 100 # constante 100
        addi $12, $0, 400 # constante 400
        addi $4, $0, 'N' # constante N
        div $8, $10
        mfhi $9

se:     bne $9, $0, fim # não é divisivel por 4?
        addi $4, $0, 'S' # constante S
        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 $4, $0, 'N' # constante N
fim:    addi $2, $0, 11
        syscall
        addi $2, $0, 10
        syscall
```

Q8) Faça um programa que leia dia, mês e ano e informe se a data é válida.

```
.test
# assumindo que os valores permitidos são maiores do que 0.
# a partir de 01/01/0000.
main:   addi $2, $0, 5
        syscall
        add $13, $2, $0 # dia
        addi $2, $0, 5
        syscall
        add $14, $2, $0 # mes
        addi $2, $0, 5
        syscall
        add $8, $2, $0 # ano
        addi $10, $0, 4 # constante 4
        addi $11, $0, 100 # constante 100
        addi $12, $0, 400 # constante 400
        addi $15, $0, 0 # constante que ao final do teste ano receberá 1 para bissexto e 0 se não for
        addi $16, $0, 12 # constante 400
        addi $17, $0, 'N' # constante N

        slt $9, $8, $0
        bne $9, $0, fim # verifica se o ano é negativo
bis:    div $8, $10
        mfhi $9
se:     bne $9, $0, mes # não é divisivel por 4?
        addi $15, $0, 1 # atualiza bissexto
```

```
        div $8, $11
        mfhi $9
senaoSe: bne $9, $0, mes # não é divisivel por 100?
        div $8, $12
        mfhi $9
senao:  beq $9, $0, mes # é divisivel por 400?
        addi $15, $0, 0 # atualiza bissexto
# Teste do mês
mes:    slt $18, $16, $14
        bne $18, $0, fim
        slt $9, $0, $14
        beq $9, $0, fim # verifica se o mes é zero ou negativo
# Teste do dia
        slt $9, $0, $13
        beq $9, $0, fim # verifica se o dia é zero ou negativo
        addi $9, $0, 2 # constante 2
        beq $14, $9, dia # testa se é fevereiro
        addi $9, $0, 30 # constante 30
        addi $10, $0, 7 # constante 7
testeM:         slt $11, $10, $14
        add $12, $14, $11
        addi $19, $0, 1 # constante 1
testeP: and $20, $19, $12 # testa se n é par ou impar
        addi $21, $20, 30
        j teste
dia:    addi $21, $15, 28
teste:  slt $22, $21, $13
        bne $22, $0, fim
        addi $17, $0, 'S' # constante S

fim:    addi $4, $17, 0
        addi $2, $0, 11
```

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

Q9) Faça um programa que leia dia, mês e ano e informe a data seguinte e a data anterior.

```
.test
# assumindo que os valores permitidos são maiores do que 0.
# a partir de 01/01/0000.
# a letra N indica que a data não é válida
main:   addi $2, $0, 5
        syscall
        add $13, $2, $0 # dia
        addi $2, $0, 5
        syscall
        add $14, $2, $0 # mes
        addi $2, $0, 5
        syscall
        add $8, $2, $0 # ano
inicio: addi $10, $0, 4 # constante 4
        addi $11, $0, 100 # constante 100
        addi $12, $0, 400 # constante 400
        addi $15, $0, 0 # constante que ao final do teste ano receberá 1 para bissexto e 0 se não for
        addi $16, $0, 12 # constante 400
        addi $17, $0, 'N' # constante N

        slt $9, $8, $0
        bne $9, $0, fim # verifica se o ano é negativo
bis:    div $8, $10
        mfhi $9
se:     bne $9, $0, mes # não é divisivel por 4?
        addi $15, $0, 1 # atualiza bissexto
        div $8, $11
```

```
        mfhi $9
senaoSe: bne $9, $0, mes # não é divisivel por 100?
        div $8, $12
        mfhi $9
senao:  beq $9, $0, mes # é divisivel por 400?
        addi $15, $0, 0 # atualiza bissexto
# Teste do mês
mes:    slt $18, $16, $14
        bne $18, $0, fim
        slt $9, $0, $14
        beq $9, $0, fim # verifica se o mes é zero ou negativo
# Teste do dia
        slt $9, $0, $13
        beq $9, $0, fim # verifica se o dia é zero ou negativo
        addi $9, $0, 2 # constante 2
        beq $14, $9, dia # testa se é fevereiro
        addi $9, $0, 30 # constante 30
        addi $10, $0, 7 # constante 7
testeM:         slt $11, $10, $14
        add $12, $14, $11
        addi $19, $0, 1 # constante 1
testeP: and $20, $19, $12 # testa se n é par ou impar
        addi $21, $20, 30
        j teste
dia:    addi $21, $15, 28
teste:  slt $22, $21, $13
        bne $22, $0, fim
        addi $23, $13, 1 # dia seguinte
        addi $9, $14, 0 # mes seguinte
        addi $10, $8, 0 # ano seguinte
        slt $24, $21, $23
        beq $24, $0, segui
```

```
        addi $23, $24, 0 # dia seguinte
        addi $9, $14, 1 # mes seguinte
        addi $11, $0, 12 # constante 12
        slt $12, $11, $9
        beq $12, $0, segui
        addi $9, $12, 0 # mes seguinte
        addi $10, $8, 1 # ano seguinte
segui:  addi $4, $23, 0
        addi $2, $0, 1
        syscall
        addi $4, $0, '/'
        addi $2, $0, 11
        syscall
        addi $4, $9, 0
        addi $2, $0, 1
        syscall
        addi $4, $0, '/'
        addi $2, $0, 11
        syscall
        addi $4, $10, 0
        addi $2, $0, 1
        syscall
        addi $4, $0, '\n'
        addi $2, $0, 11
        syscall
        # data anterior
        addi $23, $13, -1 # dia anterior
        addi $9, $14, 0 # mes anterior
        addi $10, $8, 0 # ano anterior
        slt $24, $0, $23
        bne $24, $0, anter
        addi $9, $14, -1 # mes anterior
```

```
        slt $24, $0, $9
        bne $24, $0, verifM
        addi $10, $8, -1 # ano anterior
        slt $24, $10, $0
        bne $24, $0, fim
        addi $23, $0, 31 # dia anterior
        addi $9, $0, 12 # mes anterior

verifM: addi $11, $0, 2 # constante 2
        beq $11, $9, fev # testa se é fevereiro
        addi $11, $0, 30 # constante 30
        addi $12, $0, 7 # constante 7
        slt $13, $12, $9
        add $14, $13, $9
        addi $16, $0, 1 # constante 1
        and $17, $16, $14 # testa se n é par ou impar
        addi $23, $17, 30
        j anter
fev:    addi $23, $15, 28
anter:  addi $4, $23, 0
        addi $2, $0, 1
        syscall
        addi $4, $0, '/'
        addi $2, $0, 11
        syscall
        addi $4, $9, 0
        addi $2, $0, 1
        syscall
        addi $4, $0, '/'
        addi $2, $0, 11
        syscall
        addi $4, $10, 0
```

```
        addi $2, $0, 1
        syscall
        addi $4, $0, '\n'
        addi $2, $0, 11
        syscall
        addi $2, $0, 10
        syscall

fim:    addi $4, $17, 0
        addi $2, $0, 11
        syscall
        addi $2, $0, 10
        syscall
```

Q10) Faça um programa que leia três números e informa a média aritmética simples desses três números, arredondando o resultado para o inteiro mais próximo (a partir 0,5 arredonda para o próximo inteiro maior)

```
.test
main:   addi $2, $0, 5
        syscall
        add $8, $2, $0
        addi $2, $0, 5
        syscall
        add $9, $2, $0
        addi $2, $0, 5
        syscall
        add $10, $2, $0
        addi $11, $0, 3 # constante 3
        addi $12, $0, 5 # constante 5
        add $13, $8, $9
        add $13, $13, $10 # somou os três números
div:    div $13, $11
        mfhi $14
        mflo $4
        addi $2, $0, 1
```

```
        syscall
        addi $4, $0, ','
        addi $2, $0, 11
        syscall             # imprime a parte inteira com a vírgula

        addi $15, $0, 10
        mul $17, $14, $15
        div $17, $11
        mflo $4

se:     slt $18, $4, $12
        bne $18, $0, fim
        addi $4, $4, 1
fim:    addi $2, $0, 1
        syscall

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

Q11) Um pico em uma onda mecânica é caracterizado por três valores de magnitude, *a*, *b* e *c*, tais que (*a*<*b* e *b*>*c*) forma um pico positivo ou (*a*>*b* e *b*<*c*) forma um pico negativo. Faça um programa que leia 3 números e indique se formam um pico, imprimindo a letra P, caso formem. Além disso o código deve informar se o pico é positivo negativo, acrescentando um sinal de + ou de – após a letra P. Se os três pontos não formarem um pico, deve ser impressa a letra N.

```
.test

main:   addi $2, $0, 5

        syscall

        add $8, $2, $0 # a

        addi $2, $0, 5

        syscall

        add $9, $2, $0 # b

        addi $2, $0, 5

        syscall
```

```
        add $10, $2, $0 # c
        addi $11, $0, 'N' # constante 'N'
        addi $25, $0, ' ' # constante ' '

        beq $8, $9, fim
        beq $10, $9, fim # testa se b=a ou b=c. Não formaria um pico
        slt $12, $8, $9
        slt $13, $9, $10
        add $14, $12, $13
        addi $15, $0, 1 # constante 1
        bne $14, $15, fim
        addi $11, $0, 'P' # constante 'P'
        addi $25, $0, '+' # constante '+'
se:     beq $12, $15, fim
        addi $25, $0, '-' # constante '-'

fim:    addi $4, $11, 0
        addi $2, $0, 11
        syscall
        addi $4, $25, 0
        addi $2, $0, 11
        syscall

        addi $2, $0, 10
```

```
        syscall
```

Q12) Faça um programa que leia as duas notas (entre 0 e 100) e as faltas de um aluno em uma disciplina. A média do aluno é calculada ponderadamente, com pesos 2 e 3. A cada 5 faltas o aluno perde 10 pontos. O programa deve informar a média, a penalidade e a média final, uma em cada linha.

```
.test

main:   addi $2, $0, 5

        syscall

        add $8, $2, $0 # n1

        addi $2, $0, 5

        syscall

        add $9, $2, $0 # n2

        addi $2, $0, 5

        syscall

        add $10, $2, $0 # faltas

        addi $11, $0, 3 # constante 3

        addi $12, $0, 10 # constante 10

        addi $13, $0, 5 # constante 5


        sll $14, $8, 1 # primeira nota com peso 2

        sll $15, $9, 1

        add $15, $15, $9 # segunda nota com peso 3

        add $16, $14, $15 # soma as duas notas com os pesos

divide:  div $16, $13

        mflo $4
```

```
        addi $24, $4, 0 # media final
        addi $2, $0, 1
        syscall
        addi $4, $0, '\n'
        addi $2, $0, 11
        syscall

        addi $17, $0, 0 # penalidade
        slt $18, $10, $13
        bne $18, $0, fim
        div $10, $13
        mflo $19
        mul $17, $19, $12
        sub $23, $24, $17
        slt $22, $24, $17
        beq $22, $0, fim
        addi $23, $0, 0

fim:    addi $4, $17, 0
        addi $2, $0, 1
        syscall
        addi $4, $0, '\n'
        addi $2, $0, 11
        syscall
```

```
        addi $4, $23, 0

        addi $2, $0, 1

        syscall



        addi $2, $0, 10

        syscall
```

Q13) Faça um programa que leia um número inteiro (entre 0 e 9999) e imprima esse número com 4 caracteres, sendo o número alinhado à direita.

```
.text
main:   addi $2, $0, 5
        syscall
        add $8, $0, $2 # Guarda o numero digitado em $8
        addi $9, $0, 1000 # constante 1000
        addi $10, $0, 100 # constante 100
        addi $11, $0, 10 # constante 10
        addi $12, $0, 32 # constante 32
        addi $13, $0, 11 # constante 11
        addi $14, $0, 15 # constante 15

inicio: div $8, $9
        mfhi $8 # guarda o resto
        mflo $15 # guarda o algarismo
        addi $4, $15, 48
        bne $23, $0, impM
testaM: bne $15, $0, flagM
        addi $4, $0, 32 # constante 32
        j impM
flagM:  addi $23, $0, 1 # constante 1
impM:   addi $2, $0, 11
        syscall

        div $8, $10
        mfhi $8 # guarda o resto
        mflo $15 # guarda o algarismo
        addi $4, $15, 48
        bne $23, $0, impC
testaC: bne $15, $0, flagC
        addi $4, $0, 32 # constante 32
        j impC
```

```
flagC:  addi $23, $0, 1 # constante 1
impC:   addi $2, $0, 11
        syscall

        div $8, $11
        mfhi $8 # guarda o resto
        mflo $15 # guarda o algarismo
        addi $4, $15, 48
        bne $23, $0, impD
testaD: bne $15, $0, flagD
        addi $4, $0, 32 # constante 32
        j impD
flagD:  addi $23, $0, 1 # constante 1
impD:   addi $2, $0, 11
        syscall

        addi $4, $8, 48
impU:   addi $2, $0, 11
        syscall

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

Q14) Faça um programa que leia dois números inteiros (entre 0 e 9999) e imprima esses números, um após o outro, separados por vírgula, cada um com 4 caracteres, sendo cada número alinhado à direita.

```
.text

main:   addi $2, $0, 5

        syscall

        add $8, $0, $2 # Guarda o nÃºmero digitado em $8

        addi $2, $0, 5

        syscall

        add $25, $0, $2 # Guarda o nÃºmero digitado em $25

        addi $9, $0, 1000 # constante 1000

        addi $10, $0, 100 # constante 100

        addi $11, $0, 10 # constante 10
```

```
        addi $12, $0, 32 # constante 32
        addi $13, $0, 11 # constante 11
        addi $14, $0, 15 # constante 15

inicio: div $8, $9
        mfhi $8 # guarda o resto
        mflo $15 # guarda o algarismo
        addi $4, $15, 48
        bne $23, $0, impM
testaM: bne $15, $0, flagM
        addi $4, $0, 32 # constante 32
        j impM
flagM:  addi $23, $0, 1 # constante 1
impM:   addi $2, $0, 11
        syscall

        div $8, $10
        mfhi $8 # guarda o resto
        mflo $15 # guarda o algarismo
        addi $4, $15, 48
        bne $23, $0, impC
testaC: bne $15, $0, flagC
        addi $4, $0, 32 # constante 32
        j impC
```

```
flagC:    addi $23, $0, 1 # constante 1
impC:     addi $2, $0, 11
          syscall


          div $8, $11
          mfhi $8 # guarda o resto
          mflo $15 # guarda o algarismo
          addi $4, $15, 48
          bne $23, $0, impD
testaD: bne $15, $0, flagD
          addi $4, $0, 32 # constante 32
          j impD
flagD:    addi $23, $0, 1 # constante 1
impD:     addi $2, $0, 11
          syscall


          addi $4, $8, 48
impU:     addi $2, $0, 11
          syscall


          bne $24, $0, fim # termina se imprimiu o segundo numero
          addi $4, $0, ',' # constante ,
          addi $2, $0, 11
          syscall
```

```
        addi $8, $25, 0 # atualiza $8 com o segundo numero
        addi $24, $0, 1
        addi $23, $0, 0
        j inicio # volta para imprimir o segundo numero

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

Q15) Faça um programa que leia uma data (dia, mês e ano) e informe o dia da semana (três primeiras letras) em que cai a data. Pesquise os algoritmos para fazer os cálculos.

```
.test
        # Foi usado o algoritmo de Zeller, tirado de:
        # http://programacionnerd.blogspot.com/2012/05/c-ejemplos-congruencia-de-zeller-nivel.html
main:   addi $2, $0, 5
        syscall
        add $8, $2, $0 # dia
        addi $2, $0, 5
        syscall
        add $9, $2, $0 # mes
        addi $2, $0, 5
        syscall
        add $10, $2, $0 # ano
        addi $11, $0, 12 # constante 12
        addi $12, $0, 3 # constante 3
        addi $13, $0, 1 # constante 1
```

```
if:     slt $14, $9, $12
        beq $14, $0, else
        addi $9, $9, 12 # mes + 12
        addi $10, $10, -1 # ano - 1
        j endif
else:   subi $9, $9, 2
        addi $12, $0, 100 # constante 100
endif:  div $10, $12
j:      mflo $15
k:      mfhi $16

        # Aplicando na formula:
        # h = ((700 + ((26*mes - 2)/10) + dia + k + (k/4) + ((j/4) + 5*j)) % 7);
        addi $17, $0, 26
        addi $18, $0, 10
        addi $19, $0, 5
        addi $20, $0, 7
        mul $21, $17, $9
        subi $21, $21, 2
        div $21, $18
        mflo $21 # ((26*mes - 2)/10)
        srl $22, $16, 2 # (k/4)
        srl $23, $15, 2 # (j/4)
        mul $24, $15, $19 # 5*j
```

```
        addi $25, $21, 700
        add $25, $25, $8
        add $25, $25, $16
        add $25, $25, $22
        add $25, $25, $23
        add $25, $25, $24
        # addi $25, $25, -2
        div $25, $20
h:      mfhi $25

        # imprimi dia
        addi $8, $0, 0
        addi $9, $0, 'd'
        addi $10, $0, 'o'
        addi $11, $0, 'm'

        beq $25, $8, dia # domingo
        addi $8, $8, 1
        addi $9, $0, 's'
        addi $10, $0, 'e'
        addi $11, $0, 'g'

        beq $25, $8, dia # segunda
        addi $8, $8, 1
```

```
addi $9, $0, 't'

addi $10, $0, 'e'

addi $11, $0, 'r'


beq $25, $8, dia # terca

addi $8, $8, 1

addi $9, $0, 'q'

addi $10, $0, 'u'

addi $11, $0, 'a'


beq $25, $8, dia # quarta

addi $8, $8, 1

addi $9, $0, 'q'

addi $10, $0, 'u'

addi $11, $0, 'i'


beq $25, $8, dia # quinta

addi $8, $8, 1

addi $9, $0, 's'

addi $10, $0, 'e'

addi $11, $0, 'x'


beq $25, $8, dia # sexta

addi $8, $8, 1
```

```
        addi $9, $0, 's'
        addi $10, $0, 'a'
        addi $11, $0, 'b' # sabado

dia:    addi $4, $9, 0
        addi $2, $0, 11
        syscall
        addi $4, $10, 0
        addi $2, $0, 11
        syscall
        addi $4, $11, 0
        addi $2, $0, 11
        syscall

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

Q16) *Faça um programa que leia um número com 9 algarismos e informe os dois algarismos seguintes que devem formar um CPF válido.

```
.text
main:   addi $2, $0, 5
        syscall
        add $8, $0, $2 # Guarda o numero digitado em $8
        addi $9, $0, 100000000 # constante 100000000
        addi $10, $0, 10000000 # constante 10000000
        addi $11, $0, 1000000 # constante 1000000
        addi $12, $0, 100000 # constante 100000
        addi $13, $0, 10000 # constante 10000
        addi $14, $0, 1000 # constante 1000
        addi $15, $0, 100 # constante 100
        addi $16, $0, 10 # constante 10
```

```
        addi $17, $0, 11 # constante 11
        addi $18, $0, 11 # constante 11


cpf1:   div $8, $9
        mflo $19 # guarda o algarismo
        mfhi $8
        mul $20, $19, $18 # multiplica digito 1 por 11
        add $21, $21, $20 # soma para o calculo do segundo digito
        addi $18, $18, -1
        mul $22, $19, $18 # # multiplica digito 1 por 10
        add $23, $23, $22 # soma para o calculo do primeiro digito

cpf2:   div $8, $10
        mflo $19 # guarda o algarismo
        mfhi $8
        mul $20, $19, $18 # multiplica digito 2 por 10
        add $21, $21, $20 # soma para o calculo do segundo digito
        addi $18, $18, -1
        mul $22, $19, $18 # # multiplica digito 2 por 9
        add $23, $23, $22 # soma para o calculo do primeiro digito

cpf3:   div $8, $11
        mflo $19 # guarda o algarismo
        mfhi $8
        mul $20, $19, $18 # multiplica digito 3 por 9
        add $21, $21, $20 # soma para o calculo do segundo digito
        addi $18, $18, -1
        mul $22, $19, $18 # # multiplica digito 3 por 8
        add $23, $23, $22 # soma para o calculo do primeiro digito

cpf4:   div $8, $12
        mflo $19 # guarda o algarismo
        mfhi $8
        mul $20, $19, $18 # multiplica digito 4 por 8
        add $21, $21, $20 # soma para o calculo do segundo digito
        addi $18, $18, -1
        mul $22, $19, $18 # # multiplica digito 4 por 7
        add $23, $23, $22 # soma para o calculo do primeiro digito

cpf5:   div $8, $13
        mflo $19 # guarda o algarismo
        mfhi $8
        mul $20, $19, $18 # multiplica digito 5 por 7
        add $21, $21, $20 # soma para o calculo do segundo digito
```

```
        addi $18, $18, -1
        mul $22, $19, $18 # # multiplica digito 5 por 6
        add $23, $23, $22 # soma para o calculo do primeiro digito

cpf6:   div $8, $14
        mflo $19 # guarda o algarismo
        mfhi $8
        mul $20, $19, $18 # multiplica digito 6 por 6
        add $21, $21, $20 # soma para o calculo do segundo digito
        addi $18, $18, -1
        mul $22, $19, $18 # # multiplica digito 6 por 5
        add $23, $23, $22 # soma para o calculo do primeiro digito

cpf7:   div $8, $15
        mflo $19 # guarda o algarismo
        mfhi $8
        mul $20, $19, $18 # multiplica digito 7 por 5
        add $21, $21, $20 # soma para o calculo do segundo digito
        addi $18, $18, -1
        mul $22, $19, $18 # # multiplica digito 7 por 4
        add $23, $23, $22 # soma para o calculo do primeiro digito

cpf8:   div $8, $16
        mflo $19 # guarda o algarismo
        mfhi $8
        mul $20, $19, $18 # multiplica digito 8 por 4
        add $21, $21, $20 # soma para o calculo do segundo digito
        addi $18, $18, -1
        mul $22, $19, $18 # # multiplica digito 8 por 3
        add $23, $23, $22 # soma para o calculo do primeiro digito

cpf9:   mul $20, $8, $18 # multiplica digito 9 por 3
        add $21, $21, $20 # soma para o calculo do segundo digito
        addi $18, $18, -1
        mul $22, $8, $18 # # multiplica digito 9 por 2
        add $23, $23, $22 # soma para o calculo do primeiro digito

cpf10:  div $23, $17 # soma / 11
        mflo $24
        mfhi $25
        slt $9, $25, $18 # resto < 2
        bne $9, $0, impD1 # se verdade, imprime 0
        sub $4, $17, $25 # 11 - resto
        add $19, $4, $0 # zera $4
impD1: addi $2, $0, 1
```

```
        syscall

cpf11:
        add $4, $0, $0 # zera $4
        mul $20, $19, $18 # multiplica digito 9 por 2
        add $21, $21, $20 # soma para o calculo do segundo digito
        div $21, $17 # soma / 11
        mflo $24
        mfhi $25
        slt $9, $25, $18 # resto < 2
        bne $9, $0, impD2 # se verdade, imprime 0
        sub $4, $17, $25 # 11 - resto
impD2: addi $2, $0, 1
        syscall

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

Q17) *Faça um programa que leia três inteiros correspondentes a um país de registro de um produto; um código do fabricante do produto; e um código de um produto. O programa deve imprimir o número do código verificador para um padrão de códigos de barra EAN-13. Pesquise esse padrão e veja como são feitos os cálculos.

```
.text

main:   addi $2, $0, 5

        syscall

        add $8, $0, $2 # Guarda o numero do pais

        addi $2, $0, 5

        syscall

        add $9, $0, $2 # Guarda o numero da empresa

        addi $2, $0, 5

        syscall

        add $10, $0, $2 # Guarda o numero do produto

        addi $11, $0, 10000 # constante 10000
```

```
        addi $12, $0, 1000 # constante 1000
        addi $13, $0, 100 # constante 100
        addi $14, $0, 10 # constante 10

pais:   div $8, $13
        mflo $15 # guarda o algarismo
        mfhi $8
        add $25, $25, $15 # variavel soma
        div $8, $14
        mflo $16 # guarda o algarismo
        sll $17, $16, 1
        add $17, $17, $16
        add $25, $25, $17 # variavel soma
        mfhi $18
        add $25, $25, $18 # variavel soma

empresa: div $9, $11
        mflo $15 # guarda o algarismo
        mfhi $9
        sll $16, $15, 1
        add $16, $16, $15
        add $25, $25, $16 # variavel soma
```

```
        div $9, $12
        mflo $15 # guarda o algarismo
        mfhi $9
        add $25, $25, $15 # variavel soma

        div $9, $13
        mflo $15 # guarda o algarismo
        mfhi $9
        sll $16, $15, 1
        add $16, $16, $15
        add $25, $25, $16 # variavel soma

        div $9, $14
        mflo $15 # guarda o algarismo
        mfhi $9
        add $25, $25, $15 # variavel soma

        sll $16, $9, 1
        add $16, $16, $9
        add $25, $25, $16 # variavel soma

produto: div $10, $12
        mflo $15 # guarda o algarismo
        mfhi $10
```

```
        add $25, $25, $15 # variavel soma


        div $10, $13
        mflo $15 # guarda o algarismo
        mfhi $10
        sll $16, $15, 1
        add $16, $16, $15
        add $25, $25, $16 # variavel soma


        div $10, $14
        mflo $15 # guarda o algarismo
        mfhi $10
        add $25, $25, $15 # variavel soma


        sll $16, $10, 1
        add $16, $16, $10
        add $25, $25, $16 # variavel soma


        div $25, $14
        mfhi $4
        beq $4, $0, impD
        sub $4, $14, $4


impD:   addi $2, $0, 1
```

```
        syscall


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

Q18) *Faça um programa que receba um inteiro (entre 0 e 999) e imprima o binário correspondente.

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


if:     beq $8, $0, endif
        and $25, $8, $9
        srl $8, $8, 1
        beq $8, $0, endif
        and $24, $8, $9
        srl $8, $8, 1
        beq $8, $0, endif
        and $23, $8, $9
        srl $8, $8, 1
        beq $8, $0, endif
        and $22, $8, $9
        srl $8, $8, 1
        beq $8, $0, endif
```

```
        and $21, $8, $9
        srl $8, $8, 1
        beq $8, $0, endif
        and $20, $8, $9
        srl $8, $8, 1
        beq $8, $0, endif
        and $19, $8, $9
        srl $8, $8, 1
        beq $8, $0, endif
        and $18, $8, $9
        srl $8, $8, 1
        beq $8, $0, endif
        and $17, $8, $9
        srl $8, $8, 1
        beq $8, $0, endif
        and $16, $8, $9
        srl $8, $8, 1

endif:  addi $4, $0, 0
        addi $2, $0, 1
        syscall
        addi $2, $0, 1
        syscall
        addi $2, $0, 1
```

```
syscall

addi $2, $0, 1

syscall

addi $2, $0, 1

syscall

addi $2, $0, 1

syscall

addi $2, $0, 1

syscall

addi $2, $0, 1

syscall

addi $2, $0, 1

syscall

addi $2, $0, 1

syscall

addi $2, $0, 1

syscall

addi $2, $0, 1

syscall

addi $2, $0, 1

syscall

addi $2, $0, 1

syscall

addi $2, $0, 1

syscall

addi $2, $0, 1

syscall

addi $2, $0, 1
```

```
syscall

addi $2, $0, 1

syscall

addi $2, $0, 1

syscall

addi $2, $0, 1

syscall

addi $2, $0, 1

syscall

addi $2, $0, 1

syscall

addi $2, $0, 1

syscall

addi $2, $0, 1

syscall

addi $2, $0, 1

syscall



addi $4, $16, 0

addi $2, $0, 1

syscall

addi $4, $17, 0

addi $2, $0, 1

syscall

addi $4, $18, 0

addi $2, $0, 1
```

```
        syscall
        addi $4, $19, 0
        addi $2, $0, 1
        syscall
        addi $4, $20, 0
        addi $2, $0, 1
        syscall
        addi $4, $21, 0
        addi $2, $0, 1
        syscall
        addi $4, $22, 0
        addi $2, $0, 1
        syscall
        addi $4, $23, 0
        addi $2, $0, 1
        syscall
        addi $4, $24, 0
        addi $2, $0, 1
        syscall
        addi $4, $25, 0
        addi $2, $0, 1
        syscall

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

```
        syscall
```

Q19) *Faça um programa que leia um número escrito em binário (no máximo 8 bits) e imprima seu valor em decimal.

```
.text
main:   addi $2, $0, 5
        syscall
        add $8, $0, $2
        addi $9, $0, 10 # constante 10
        addi $10, $0, 1 # constante 1

if:     beq $8, $0, endif
        div $8, $9
        mflo $8
        mfhi $11
        mul $12, $11, $10
        add $4, $4, $12
        sll $10, $10, 1

        beq $8, $0, endif
        div $8, $9
        mflo $8
        mfhi $11
        mul $12, $11, $10
        add $4, $4, $12
        sll $10, $10, 1
```

```
beq $8, $0, endif

div $8, $9

mflo $8

mfhi $11

mul $12, $11, $10

add $4, $4, $12

sll $10, $10, 1


beq $8, $0, endif

div $8, $9

mflo $8

mfhi $11

mul $12, $11, $10

add $4, $4, $12

sll $10, $10, 1


beq $8, $0, endif

div $8, $9

mflo $8

mfhi $11

mul $12, $11, $10

add $4, $4, $12

sll $10, $10, 1
```

```
beq $8, $0, endif

div $8, $9

mflo $8

mfhi $11

mul $12, $11, $10

add $4, $4, $12

sll $10, $10, 1


beq $8, $0, endif

div $8, $9

mflo $8

mfhi $11

mul $12, $11, $10

add $4, $4, $12

sll $10, $10, 1


beq $8, $0, endif

div $8, $9

mflo $8

mfhi $11

mul $12, $11, $10

add $4, $4, $12

sll $10, $10, 1
```

```
endif:  addi $2, $0, 1
        syscall


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

Q20) *Dizemos que um número *i* é congruente módulo *m* a *j* se *i* % *m* = *j* % *m*. Exemplo: 35 é congruente módulo 4 a 39, pois 35 % 4 = 3 = 39 % 4.
Faça um programa que, dados i, j e m, informe se i e j são congruentes módulos m

```
.text
main:   addi $2, $0, 5
        syscall
        add $8, $0, $2 # Guarda o numero de i
        addi $2, $0, 5
        syscall
        add $9, $0, $2 # Guarda o numero de j
        addi $2, $0, 5
        syscall
        add $10, $0, $2 # Guarda o numero de m

if:     div $8, $10
        mfhi $11
        div $9, $10
        mfhi $12

        addi $4, $0, 'S'
        beq $11, $12, endif
        addi $4, $0, 'N'

endif:  addi $2, $0, 11
```

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

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