E-Fólio A: **Divisibilidade por 4 e Não por 8 em Diferentes Bases**

1. Números Inteiros em Notação Decimal

Lógica e Explicação

Um número inteiro decimal (N) é um múltiplo de 4 **se e somente se** o número formado pelos seus últimos dois dígitos (b) for um múltiplo de 4.

Para que N seja um múltiplo de 4, mas **não** um múltiplo de 8, precisamos considerar duas condições que dependem do valor de b :

1. ***N* é Múltiplo de 4:** Os últimos dois dígitos, b, devem pertencer ao conjunto: $M_4 = \{00,04,08,12,16,20,24,28,32,36,40,44,48,52,56,60,64,68,72,76,80,84,88,92,96\}$
2. ***N* é NÃO Múltiplo de 8:** N não pode ser divisível por 8. A divisibilidade por 8 é determinada pela fórmula $N = 100a + b$, onde a são os dígitos que precedem b. Como $100 = 12 \times 8 + 4$, a divisibilidade de N por 8 depende da paridade de a e do resto de b por 8.

Isto leva a duas regras completas para $\mathrm{N} \equiv 4 \pmod 8$ (ou seja, múltiplo de 4, mas não de 8):

- **Regra A (Múltiplos de 4 que não são Múltiplos de 8):** O número b é da forma $8k + 4$ (resto 4 na divisão por 8).
    - $b \in \{04, 12, 20, 28, 36, 44, 52, 60, 68, 76, 84, 92\}$
    - O número de centenas (a) deve ser **par** (ou 0).
- **Regra B (Múltiplos de 8):** O número b é da forma $8k$ (resto 0 na divisão por 8).
    - $b \in \{00, 08, 16, 24, 32, 40, 48, 56, 64, 72, 80, 88, 96\}$
    - O número de centenas (a) deve ser **ímpar**.

A expressão regular concentra-se nos **últimos dois dígitos** (b), assumindo que N é um número positivo:

$$\text{Regra:} \quad N = \cdots d_1 d_0$$

$$\text{Onde:} \quad d_1 d_0 \in \{04,12,20,28,36,44,52,60,68,76,84,92\}$$

Prova formal:

Seja (N) um inteiro positivo qualquer. Podemos escrever (N) da seguinte forma:
$N = 100a + b$

onde (b) é o número formado pelos dois últimos algarismos $(0 \leq b \leq 99)$ e $100a$ é o número formado por todos os algarismos anteriores (centenas, milhares, etc.).

Sabemos que: $100 = 8 \times 12 + 4$. Assim, podemos reescrever (N) como:

$$N = a(100) + b = a(8 \times 12 + 4) + b = (8 \times 12a) + (4a + b)$$

É múltiplo de 4?

$$N = 4(25a) + b$$

Como $4(25a)$ é sempre um múltiplo de 4, conclui-se que (N) é múltiplo de 4 se e só se (b) for múltiplo de 4. (Esta é a regra habitual de divisibilidade por 4.)

É múltiplo de 8?

$$N = (8 \times 12a) + (4a + b)$$

Como $8 \times 12a$ é sempre múltiplo de 8, conclui-se que (N) é múltiplo de 8 se e só se $(4a + b)$ for múltiplo de 8.

A condição: múltiplo de 4 mas não de 8

Condição 1 (Múltiplo de 4): (b) tem de pertencer ao conjunto dos múltiplos de 4, M_4.

Condição 2 (Não múltiplo de 8) : $(4a + b)$ não pode ser múltiplo de 8.

Consideram-se dois casos para $(b \in M_4)$:

Caso A: (b) é múltiplo de 8 (por exemplo: $(00, 08, 16, 24, \ldots)$

Se $b = 8k$, então: $4a + b = 4a + 8k$

A expressão $4a + b$ é múltiplo de 8 se e só se $4a$ for múltiplo de 8, o que acontece quando (a) é um número par.

Se (a) for ímpar, então (N) será múltiplo de 4 mas não de 8.

Exemplo:

$$a = 1, \quad b = 08$$

$$N = 108$$

$$108 / 4 = 27$$

$$(108 / 8 = 13 \, , resto \, 4$$

Caso B: (b) não é múltiplo de 8 (por exemplo: ($04, 12, 20, 28, \ldots$))

Se $b = 8k + 4$, então: $4a + b = 4a + 8k + 4 = 4(a + 1) + 8k$

A expressão $4(a + 1) + 8k$ é múltiplo de 8 se e só se $4(a + 1)$ for múltiplo de 8, o que acontece quando $(a + 1)$ é par.

- Se $(a + 1))$ é par, então (a) é ímpar, e (N) é múltiplo de 8.
- Se $(a + 1)$ é ímpar, então (a) é par, e (N) não é múltiplo de 8.

Exemplo:

$$a = 2, b = 12$$

$$N = 212$$

$$212 / 4 = 53$$

$$212 / 8 = 26, resto\ 4$$

Se os dois últimos algarismos pertencem à lista $\{04, 12, 20, 28, 36, 44, 52, 60, 68, 76, 84, 92\}$, então o número é sempre múltiplo de 4. Para que **não seja múltiplo de 8**, é ainda necessário que o algarismo das centenas seja **par ou inexistente**.

Como a lista é, $\{04, 12, 20, 28, 36, 44, 52, 60, 68, 76, 84, 92\}$ temos $b = 8k + 4$ logo , Se (b) pertence à lista, então (N) é sempre múltiplo de 4. Para que (N) seja múltiplo de 8, o algarismo das centenas (a) tem de ser ímpar. Para que (N) não seja múltiplo de 8, o algarismo das centenas tem de ser par ou $a = 0$. Todos os números de dois algarismos da lista são múltiplos de 4 mas não de 8.

Expressão Regular (UNIX)

Em notação UNIX, é possível construir uma expressão regular válida para os inteiros decimais que são múltiplos de 4 mas não de 8. No entanto, essa expressão não realiza cálculo aritmético; ela codifica explicitamente as consequências estruturais do critério de divisibilidade, exigindo a separação de casos e enumeração de padrões.

Expressão (considerando a paridade de *a*):

Esta expressão é complexa e ultrapassa o limite da notação UNIX simples, pois a verificação de paridade de a (os dígitos que precedem os últimos dois) exige *lookarounds* (que não existem na notação básica) ou uma enumeração exaustiva de padrões:

Padrão 1 (Últimos 2 dígitos $8k + 4$ e a par)

Padrão 2 (Últimos 2 dígitos $8k$ e a ímpar)

Como as expressões regulares UNIX simples não conseguem verificar a paridade de um número arbitrariamente longo de dígitos,

Vamos assumir e problema e assumir que a expressão será construída para:

- Para números com 1 ou 2 algarismos, basta olhar para o próprio número.
- Para números com 3 ou mais algarismos, a divisibilidade por 8 depende do algarismo das centenas.

Assim, criamos três grandes blocos no regex.

Últimos dois dígitos da forma

$$b = 8k + 4$$

Caso 1 — números com 1 ou 2 algarismos

Todos estes são automaticamente:

- múltiplos de 4
- não múltiplos de 8

Regex: $^(4|12|20|28|36|44|52|60|68|76|84|92)\$$

Caso 2 — números com 3 ou mais algarismos (algarismo das centenas par)

Aqui forçamos explicitamente:

- qualquer prefixo
- um dígito par imediatamente antes dos últimos dois
- últimos dois dígitos ≡ 4 (mod 8)

Regex: $^[0-9]*[02468](04|12|20|28|36|44|52|60|68|76|84|92)\$$

Caso 3 — números com 3 ou mais algarismos (um dígito ímpar imediatamente antes dos últimos dois)

Aqui forçamos explicitamente:

- qualquer prefixo
- um dígito par imediatamente antes dos últimos dois
- últimos dois dígitos ≡ 4 (mod 8)

Regex: $^[0-9]*[13579](00|08|16|24|32|40|48|56|64|72|80|88|96)$

Expressão (união dos casos)

$$^{(}\ (4|12|20|28|36|44|52|60|68|76|84|92)\ |\ [0-9]$$
$$*\ [02468](04|12|20|28|36|44|52|60|68|76|84|92)\ |\ [0-9]$$
$$*\ [13579](00|08|16|24|32|40|48|56|64|72|80|88|96))\$$

Esta expressão regular não 'calcula' divisibilidade; ela codifica diretamente as consequências aritméticas do critério de divisibilidade por 4 e 8, respeitando as limitações da notação regex UNIX.

Explicação da Construção (Expressão Simplificada):

- ^: Início da linha (garante que estamos a procurar a partir do início do número).
- (4|12|20|28|36|44|52|60|68|76|84|92)
  Primeiro caso: números com 1 ou 2 algarismos.
- **|** Operador de **alternativa lógica (OU)**.
- ([0-9]*): Captura **zero ou mais** dígitos (0-9).
- (04|12|20|28|36|44|52|60|68|76|84|92): Um grupo de captura que verifica se os últimos dois dígitos correspondem exatamente a um dos 12 pares que são $\equiv 4\ (mod\ 8)$.
- [02468] = garante **a par**
- (04|12|20|28|36|44|52|60|68|76|84|92) Grupo que corresponde exatamente aos dois últimos algarismos do número.
- $: Fim da linha (garante que os pares de dígitos acima são o final do número).

Finalmente permitimos negativos usando -? Permite zero ou uma ocorrência do sinal negativo e positivo.

Expressão Final:

$$^{[+-]?(}\ (4|12|20|28|36|44|52|60|68|76|84|92)\ |\ [0-9]$$
$$*\ [02468](04|12|20|28|36|44|52|60|68|76|84|92)\ |\ [0-9]$$
$$*\ [13579](00|08|16|24|32|40|48|56|64|72|80|88|96))\$$

2. Números Inteiros em Notação Binária

Lógica e Explicação

Em notação binária, números negativos são representados exclusivamente por bits (0 e 1), tipicamente em complemento para dois. Contudo, a interpretação do sinal depende do comprimento da palavra binária. Como expressões regulares não têm noção do bit mais significativo nem do comprimento total da palavra, não é trivial, em geral, distinguir números binários positivos e negativos nem verificar propriedades aritméticas em complemento para dois sem fixar previamente o número de bits.

Então vamos ter que assumir opções, uma opção é assumir apenas números positivos, seguiremos para já assumindo essa limitação.

Em notação binária (base 2), a divisibilidade por potências de 2 é extremamente simples:

- Múltiplo de 4 (2^2): Um número binário é divisível por 2^k se e só se terminar em k zeros. Como $4 = 2^2$, o número tem que terminar em 00.

$$N = (\dots d_2 d_1 00)_2$$

- NÃO Múltiplo de 8 (2^3): Um número binário é divisível por $8 = 2^3$ se e só se terminar em 000. Para **não** ser um múltiplo de 8, o número **não** pode terminar em 000.

Seja Num inteiro positivo qualquer. Quando Né escrito em binário, podemos representá-lo da seguinte forma:

$$N = (\dots d_3 d_2 d_1 d_0)_2$$

onde $d_i \in \{0,1\}$são os algarismos binários, e:

$$N = \cdots + d_3 \cdot 2^3 + d_2 \cdot 2^2 + d_1 \cdot 2^1 + d_0 \cdot 2^0$$

Múltiplo de 4 (2^2):

Um número é divisível por 2^k(como $4 = 2^2$) **se e só se** os seus últimos $k$bits forem zero.

Como $4 = 2^2$, temos $k = 2$.

Assim, Né múltiplo de 4 se os dois últimos bits forem 00:

$$N = (\dots d_3 d_2 \mathbf{00})_2$$

Isto acontece porque:

$$N = (4 \times \text{um certo inteiro}) + (\text{os dois últimos bits})$$

e, se os dois últimos bits forem 00, o resto da divisão por 4 é zero.

Não múltiplo de 8 (2^3):

Um número é divisível por 2^k(como $8 = 2^3$) **se e só se** os seus últimos $k$bits forem zero.

Como $8 = 2^3$, temos $k = 3$.

Logo, Né múltiplo de 8 se os três últimos bits forem 000.

Consequentemente, N**não** é múltiplo de 8 se os três últimos bits **não** forem 000.

Combinação das condições:

Para que um número seja múltiplo de 4 mas **não** múltiplo de 8, tem de satisfazer simultaneamente as duas regras:

Regra 1: Terminar em $\mathbf{00}$

$$N = (\dots d_3\mathbf{00})_2$$

Regra 2: Não terminar em $\mathbf{000}$

Combinando estas condições, os três últimos algarismos binários têm de ser:

d_2	d_1	d_0	Condições
?	$\mathbf{0}$	$\mathbf{0}$	Satisfaz a Regra 1 (múltiplo de 4)
$\mathbf{1}$	$\mathbf{0}$	$\mathbf{0}$	Satisfaz a Regra 2 (não é múltiplo de 8, pois $100 \neq 000$)

A única forma de terminar em 00 mas não terminar em 000é o terceiro algarismo a contar do fim (d_2) ser $\mathbf{1}$.

Assim, em binário, o número tem de terminar **exatamente** em $\mathbf{100}$.

Esta regra em binário explica de forma elegante a lista decimal apresentada anteriormente: $04,12,20,28,$ … Vejamos alguns números decimais e as suas representações binárias:

DECIMAL	BINÁRIO	TERMINA EM	DIVISÍVEL POR 4?	DIVISÍVEL POR 8?
4	100	**100**	Sim	Não
12	1100	**100**	Sim	Não
20	10100	**100**	Sim	Não
28	11100	**100**	Sim	Não
36	100100	**100**	Sim	Não

Expressão Regular (UNIX)

$$\text{^}[10]*100\$$$

Explicação da Construção:

- ^: Início da linha.
- [10]*: Zero ou mais ocorrências dos dígitos binários 1 ou 0. (Isto abrange qualquer comprimento para os dígitos que antecedem o padrão).
- 100: O padrão obrigatório. O número deve terminar exatamente com os dígitos 100 (garantindo que é múltiplo de 4 e não é múltiplo de 8).
- $: Fim da linha.

Vamos agora assumir números negativos, considerando sempre as limitações referidas anteriormente. Uma outra opção que poderíamos tomar é de se fixar o número de bits, então já é possível proseguir.

Exemplo: 8 bits, complemento para dois: $\text{^}[01]\{5\}100\$$

Explicação:

- os 3 últimos bits 100 → múltiplo de 4, não de 8
- os 5 bits iniciais podem ser qualquer coisa
- MSB = 1 ⇒ negativo, MSB = 0 ⇒ positivo

⚠ **Mas só para 8 bits**

3. Números Inteiros em Notação Hexadecimal

Lógica e Explicação

Em notação hexadecimal, números negativos são representados por complemento para dois, sendo o sinal determinado pelo bit mais significativo. Como o significado desse bit depende do comprimento total da palavra, expressões regulares não conseguem trivialmente, em geral, distinguir números positivos e negativos nem verificar propriedades aritméticas em hexadecimal sem que o número de dígitos seja previamente fixado.

Em hexadecimal puro, tal como no binário:

- não existe sinal "–"
- números negativos são representados apenas por dígitos hexadecimais

- usando uma codificação, normalmente complemento para dois

Hexadecimal é apenas uma representação compacta do binário: cada dígito hex = 4 bits

Logo, tudo o que foi dito para o binário aplica-se integralmente a hexadecimal.

Então seguiremos com as mesmas presunções anteriores.

Em notação hexadecimal (base 16), a divisibilidade por 4 e 8 é determinada apenas pelo valor do **último dígito** hexadecimal.

- Múltiplo de 4: O valor decimal do último dígito deve ser um múltiplo de 4.

$$\text{Dígitos Múltiplos de } 4: \quad \{0,4,8,C\}$$

- NÃO Múltiplo de 8: O valor decimal do último dígito não pode ser um múltiplo de 8.

$$\text{Dígitos Múltiplos de } 8: \quad \{0,8\}$$

No sistema hexadecimal (base 16), os algarismos vão de 0a 9e depois de Aaté F(representando os valores decimais de 10a 15).

A lógica é semelhante à do sistema binário, mas como 4e 8são potências de 2, e 16também é uma potência de 2, a regra de divisibilidade envolve a observação do **último algarismo hexadecimal**.

Múltiplo de 4:

Um número é múltiplo de $16/4 = 4$ **se e só se** o seu último algarismo hexadecimal representar um número que seja múltiplo de 4. Os algarismos hexadecimais (e os seus valores decimais) que são múltiplos de 4 são:

- 0(decimal 0)
- 4(decimal 4)
- 8(decimal 8)
- C(decimal 12)

Isto significa que o último algarismo tem de ser par $(0,2,4,6,8,A,C,E)$, mas, mais precisamente, tem de ser **divisível por 4**, ou seja, $0,4,8$ou C.

A regra "o último algarismo é par" é tecnicamente verdadeira, porque todo o múltiplo de 4 é par, mas a regra correta e exata em hexadecimal é: "o último algarismo é $0,4,8$ ou C".

Não múltiplo de 8:

Para que um número seja múltiplo de 8, o seu último algarismo hexadecimal tem de representar um número que seja múltiplo de 8. Os algarismos hexadecimais que são múltiplos de 8 são:

- 0(decimal 0)
- 8(decimal 8)

Assim, para que um número **não** seja múltiplo de 8, o seu último algarismo hexadecimal **não** pode ser 0 nem 8.

Combinação das condições:

Para que um número seja múltiplo de 4 mas **não** múltiplo de 8, o último algarismo hexadecimal tem de ser um múltiplo de 4, excluindo os múltiplos de 8.

Condição	***Algarismos hexadecimais (valor decimal)***
Múltiplo de 4	$0,4,8,C$(0, 4, 8, 12)
NÃO múltiplo de 8	**Não pode ser** 0nem 8

Tomando a intersecção:

$$\{0,4,8,C\} \setminus \{0,8\} = \{4,C\}$$

Os dígitos hexadecimais válidos são **4** e **C**.

$$\text{Padrão Hexadecimal:} \quad \ldots d_0$$

$$\text{Onde:} \quad d_0 \in \{4,C\}$$

Verificação com congruências (módulo):

A condição final — *"o último algarismo* $\equiv 4(\text{mod}8)$*"* — é a forma mais formal de expressar esta regra: O valor decimal do último algarismo hexadecimal tem de deixar resto 4 quando dividido por 8.

Algarismo hex.	***Valor decimal***	***Valor decimal (mod 8)***	***Resultado***
4	4	$4(\text{mod}8)$	4 (válido)
C	12	$12(\text{mod}8)$	4 (válido)
0	0	$0(\text{mod}8)$	0
8	8	$8(\text{mod}8)$	0

Os dois algarismos cujos valores decimais são congruentes com $4(\mathrm{mod}8)$ são, de facto, **4 e C**.

Padrão consistente entre bases numéricas

Isto revela um padrão coerente entre diferentes bases:

- **Decimal:** os dois últimos algarismos têm de ser $4(\mathrm{mod}8)$(por exemplo, $04,12,20,$ ...)
- **Binário:** o número tem de terminar em 100(que representa o valor 4)
- **Hexadecimal:** o último algarismo tem de ser 4ou C (valores 4 e 12, ambos $4(\mathrm{mod}8)$)

Expressão Regular (UNIX)

$$\text{^}[0-9A-Fa-f]*[4Cc]\$$$

Explicação da Construção:

- ^: Início da linha.
- [0-9A-Fa-f]*: Zero ou mais ocorrências de dígitos hexadecimais (0-9 e letras A-F, tanto maiúsculas quanto minúsculas, para garantir robustez).
- [4Cc]: O padrão obrigatório para o último dígito. O número deve terminar com um dígito que seja 4 o C (considerando a possibilidade de ser minúsculo c).
- $: Fim da linha.

Agora vamos considerar um exemplo de negativos. Uma outra opção que poderíamos tomar é de se fixar o complemento para dois com comprimento fixo.

Exemplo: **32 bits (8 dígitos hex)**

$$\text{^}[0-9A-Fa-f]\{7\}[4Cc]\$$$

Agora:

- os 7 primeiros dígitos definem o sinal
- o último nibble garante divisibilidade

⚠ **Apenas para 32 bits**