






Uma caixa de clips contém 0223_4 (base 4) clips amarelos e $0x0d$ (base 16) clips vermelhos. Pretende-se calcular o total de clips disponíveis. Siga os passos abaixo.

Usando sempre uma representação em binário com 7 bits, calcule o total de clips, em binário, mostrando as duas parcelas (primeiro os clips amarelos, depois os clips vermelhos) seguido do resultado (primeiro em binário com 7 bits, depois em octal com 3 dígitos):

$$[OP1]_{-2} + [OP2]_{-2} = [RB]_{-2} = [RH]_{-8}$$


Resposta Especificada para OP1  0101011

Resposta Especificada para OP2  0001101

Resposta Especificada para RB  0111000

Resposta Especificada para RH  070

Respostas Corretas para OP1

Método de avaliação	Resposta Correta	Diferenciação de maiúsculas e minúsculas
 Correspondência Exata	0101011	


Respostas Corretas para OP2


Método de avaliação	Resposta Correta	Diferenciação de maiúsculas e minúsculas
 Correspondência Exata	0001101	

Respostas Corretas para RB

Método de avaliação	Resposta Correta	Diferenciação de maiúsculas e minúsculas
 Correspondência Exata	0111000	

Respostas Corretas para RH

Método de avaliação	Resposta Correta	Diferenciação de maiúsculas e minúsculas
 Correspondência de padrão	\s*0*70	

 Considere as seguintes sequências de 7 bits, apresentadas em octal: 106_8 , 067_8 .

Apresente o binário correspondente (com 7 bits, sem espaços):


$$106_8 = [B1]$$


$$067_8 = [B2]$$


Sabendo que cada um desses valores representa um inteiro em complemento para 2, adicione os dois valores e apresente o resultado em binário (com 7 bits, sem espaços):


$$106_8 + 067_8 = [B3]$$

Apresente o resultado da alínea anterior em binário usando 7 bits (sem espaços) e representação em sinal + amplitude: [E]

Resposta Especificada para B1  1000110

Resposta Especificada para B2  0110111

Resposta Especificada para B3  1111101

Resposta Especificada para E  1000011

Respostas Corretas para B1

Método de avaliação

 Correspondência Exata

Resposta Correta

1000110

Diferenciação de maiúsculas e minúsculas

Respostas Corretas para B2

Método de avaliação

 Correspondência Exata

Resposta Correta

0110111

Diferenciação de maiúsculas e minúsculas

Respostas Corretas para B3

Método de avaliação

 Correspondência Exata

Resposta Correta

1111101

Diferenciação de maiúsculas e minúsculas

Respostas Corretas para E

Método de avaliação

 Correspondência Exata

Resposta Correta

1000011

Diferenciação de maiúsculas e minúsculas



Considere um formato de representação de números em vírgula flutuante, baseado na norma IEEE 754 com 12 bits: 1 bit para sinal, 6 bits para o expoente e 5 bits para a parte fracionária.

Represente neste formato (em binário e sem espaços no meio) o valor $+25.8125 \times 64$.

Nota: Se forem necessários mais de 5 bits para representar a parte fracionária, utilize truncagem do valor.

Valor em binário = 0b **[valbin]**

Resposta Especificada para valbin 010100110011

Respostas Corretas para valbin

Método de avaliação

Correspondência de padrão

Resposta Correta

`\s*010100110011\s*`

Diferenciação de maiúsculas e minúsculas



Considere um formato de representação de números em vírgula flutuante, baseado na norma IEEE 754, com 12 bits: 1 bit para sinal, 6 bits para o expoente e 5 bits para a parte fracionária.

Calcule o valor na base 10 que corresponde ao valor 0x42D escrito naquele formato. O resultado deverá ser apresentado como: +/- valor_base10_com_ponto_decimal (sem espaços e com os dígitos estritamente necessários)
Valor na base 10 = **[decimal]**

Resposta Especificada para decimal +5.625

Respostas Corretas para decimal

Método de avaliação

Correspondência de padrão

Correspondência de padrão

Resposta Correta

`\s*(?i)(\+?\s*5.625)\s*`

`\s*(?i)(\+?\s*5,625)\s*`

Diferenciação de maiúsculas e minúsculas



O estado (parcial) de um sistema com um processador IA-16 (*little endian*) é descrito por:

- conteúdo de alguns registos:

`%ip = 0x6514`

`%ax = 0x0010`

`%sp = 0x77b2`

`%bp = 0xa8a6`

- conteúdo de algumas células de memória, em hexadecimal, a partir de cada um dos endereços:

`0x6508: 05 00 ef ff 01 1a 24 1f`

`0x6510: 1a 02 5f 17 ab 23 7f 2b`

...

`0x13a8: 43 f8 ff 01 43 2b 45 23`

`0x13b0: ef ff 01 98 23 e5 c3 23`

O processador vai ler a próxima instrução a ser executada.

Mostre o bloco de informação que circula em cada um dos seguintes barramentos, em hexadecimal:

- barramento de endereços: `0x[A]`

- barramento de dados: `0x[B]`

Resposta Especificada para A 6514

Resposta Especificada para B ab

Respostas Corretas para A

Método de avaliação

Correspondência Exata

Resposta Correta

6514

Diferenciação de maiúsculas e minúsculas

Respostas Corretas para B


Método de avaliação

Correspondência de padrão

Resposta Correta

(?i)(ab23)(?i)(23ab)

Diferenciação de maiúsculas e minúsculas

 O estado (parcial) de um sistema com um processador IA-16 (little endian) é descrito por:

- conteúdo de alguns registros:

`%ip = 0x210e`

`%ax = 0x0010`

`%sp = 0x56ac`

`%bp = 0x56ae`

- conteúdo de algumas células de memória, em hexadecimal, a partir de cada um dos endereços:

`0x3408: 05 00 ef ff 01 1a 24 1f`

`0x3410: 1a 02 5f 17 ab 23 7f 2b`

`...`

`0x56a8: 43 f8 ff 01 43 2b 45 23`

`0x56b0: ef ff 01 98 23 e5 c3 23`

O processador acabou de decodificar a instrução e vai iniciar a sua execução:

- `push %bp`

1.) Mostre o bloco de informação que circula em cada um dos seguintes barramentos, em hexadecimal:

- barramento de endereços: **0x[A]**

- barramento de dados: **0x[B]**

Resposta Especificada para A  56aa


Resposta Especificada para B  56ae

Respostas Corretas para A

Método de avaliação	Resposta Correta	Diferenciação de maiúsculas e minúsculas
 Correspondência Exata	56aa	

Respostas Corretas para B

Método de avaliação	Resposta Correta	Diferenciação de maiúsculas e minúsculas
 Correspondência Exata	56ae	

 Considere o estado parcial de um PC com um processador IA-16 (*little endian*), ilustrado abaixo.

Registos	Memória (código)	Memória (dados)
%ip = 0x4160	...	0x6ffc: 0x50 0x00
%bp = 0x7008	0x4159 incw %ax	0x6ffe: 0x10 0x33
%ax = 0x5599	0x415a movw 4(%bp), %bx	0x7000: 0xb5 0x40
%cx = 0x0032	0x4160 popw %ax	0x7002: 0x20 0x00
%sp = 0x7000	0x4161 addw %cx, -4(%bp)	0x7004: 0x11 0x22
	0x4164 incw %ax	0x7006: 0x01 0x00
	0x4165 ...	0x7008: 0xff 0x00

Após a execução das 2 próximas instruções, indique o conteúdo dos seguintes:

- registos**

%ip = 0x[rip]

%sp = 0x[rsp]

- células de memória:**

0x7003: 0x[rm1]

0x7004: 0x[rm2]

Resposta Especificada para rip  4164

Resposta Especificada para rsp  7002

Resposta Especificada para rm1  00

Resposta Especificada para rm2  43

Respostas Corretas para rip

Método de avaliação	Resposta Correta	Diferenciação de maiúsculas e minúsculas
 Correspondência de padrão	<code>\s*4164\s*</code>	

Respostas Corretas para rsp

Método de avaliação	Resposta Correta	Diferenciação de maiúsculas e minúsculas
 Correspondência de padrão	<code>\s*(?i)(7002)\s*</code>	

Respostas Corretas para rm1

Método de avaliação	Resposta Correta	Diferenciação de maiúsculas e minúsculas
 Correspondência de padrão	<code>\s*(?i)(00)\s*</code>	

Respostas Corretas para rm2

Método de avaliação	Resposta Correta	Diferenciação de maiúsculas e minúsculas
 Correspondência de padrão	<code>\s*(?i)(43)\s*</code>	

 Considere o código *assembly* seguinte:




```
leal (%eax,%ecx,4), %ebx
addl %ebx, (%eax)
```




Indique se as seguintes afirmações são verdadeiras (V) ou falsas (F):


A execução de cada instrução implica pelo menos um acesso à memória. **[r1]**

A instrução `leal` calcula o endereço `[%ecx + %eax * 4]`. **[r2]**

O registo `%ebx` é alterado na execução deste programa. **[r3]**



- Resposta Especificada para r1  F
- Resposta Especificada para r2  F
- Resposta Especificada para r3  V

Respostas Corretas para r1		
Método de avaliação	Resposta Correta	Diferenciação de maiúsculas e minúsculas
 Correspondência de padrão	<code>\s*\(*(ff)[aA]*[iL]*[sS]*[oO]*\)*\s*</code>	
Respostas Corretas para r2		
Método de avaliação	Resposta Correta	Diferenciação de maiúsculas e minúsculas
 Correspondência de padrão	<code>\s*\(*(ff)[aA]*[iL]*[sS]*[oO]*\)*\s*</code>	
Respostas Corretas para r3		
Método de avaliação	Resposta Correta	Diferenciação de maiúsculas e minúsculas
 Correspondência de padrão	<code>\s*\(*(vv)[eE]*[rR]*[dD]*[aA]*[dD]*[eE]*[iI]*[rR]*[oO]*\)*\s*</code>	


 Considere a seguinte função em C, compilada para IA-32, e um fragmento do resultado da sua compilação para *assembly*.

Complete o código *assembly* preenchendo os campos em falta.

Função C	Assembly
<pre>int prod_s_oposto (int x, int y) { return (x + y) * (x - y); }</pre>	<pre>... mov 0x8(%ebp),%ebx mov 0xc(%ebp),%edx lea ([ra] , %ebx ,1) , %eax sub %edx, [rb] imul %ebx, %eax ...</pre>

- Resposta Especificada para ra  %edx
- Resposta Especificada para rb  %ebx

Respostas Corretas para ra		
Método de avaliação	Resposta Correta	Diferenciação de maiúsculas e minúsculas
 Correspondência de padrão	<code>\s*%edx\s*</code>	
Respostas Corretas para rb		
Método de avaliação	Resposta Correta	Diferenciação de maiúsculas e minúsculas
 Correspondência de padrão	<code>\s*%ebx\s*</code>	

 Identifique a função (escrita em C) que originou as seguintes instruções:

```
pushl    %ebp
movl     %esp, %ebp
pushl    12(%ebp)           # coloca na
pilha o  que está na memória em 12(%ebp)
call     perimetroCircunferencia
leave
ret
```

Respostas Seleccionadas:

```
int f1(int l, int r, int f, int m){
    perimetroCircunferencia(r);
}
```

 b.

Respostas:

```
int f1(int l, int r, int f, int m){
    perimetroCircunferencia(l);
}
```

a.

```
int f1(int l, int r, int f, int m){
    perimetroCircunferencia(r);
}
```


 b.

```
int f1(int l, int r, int f, int m){
    perimetroCircunferencia(f);
}
```

c.

```
int f1(int l, int r, int f, int m){
    perimetroCircunferencia(m);
}
```

d.

 Num sistema de computação o processador (IA-32) acede ao estado dum periférico através de um seu “*registo de estado*”, onde cada bit tem um significado específico para esse periférico.

Considere que o conteúdo desse registo já foi copiado para o registo `%ecx` do processador e que se pretende **ler e alterar** o seu **bit 2** (o 3º bit a contar da direita), e apenas este bit (os restantes não devem ser modificados).

Escreva código *assembly* para realizar apenas estas 2 operações:

- ler apenas esse bit, colocando o seu valor como *True* ou *False* (i.e., 0 ou 1) no registo `%al`; e
- **fazer o *toggle* desse bit** (*toggle*: se estiver a zero passa para um, se estiver a um passa para zero), colocando o resultado no registo `%cl`.

Resposta Selecionada: `test $0x01, %eax` # Vamos fazer um bit-wise comparison de modo a compararmos apenas o 2º bit, desta forma sabemos se ele será 1 ou 0 (*True* ou *False*)
`sete %al` # Desta forma `%al` = resultado lógico (0/1) da comparação
`subl $0x01, %eax` # Caso o 2º bit seja 1 iremos subtrair 0x01 de modo a que o 2º bit fique a 0

Resposta Correta: [Nenhuma]



As operações com valores reais em FP com meia-precisão (com 16 bits) podem ser executadas em metade do tempo de operação de valores com precisão simples (com 32 bits), embora estes valores podem apenas representar gamas de valores mais reduzidas e com menor precisão.

Contudo, há 2 formatos para o representar: um que segue a recomendação IEEE-754 (1 bit para o sinal, 5 bits para o expoente e 10 bits para a parte fracionária), e outro desenvolvido pela Google e com muito boa aceitação dos fabricantes de componentes para processarem informação, o `bf16` (1 bit para o sinal, 8 bits para o expoente e 7 bits para a parte fracionária).

Como explica a popularidade do `bf16` em relação ao formato recomendado pela IEEE-754?

Resposta Selecionada: Como dado pelo enunciado o `bf16` tem menos algarismos significativos (7 bits comparativamente aos 10 bits usados pelo IEEE-754) e tem um aumento de bits para o expoente (8 bits comparativamente aos 5 bits usados pelo IEEE-754).

Imagino que este formato seja popular para processar informação (Big Data), especialmente para casos como o de machine learning visto que papers relativamente recentes dizem-nos que mesmo diminuindo o numero de algarismos significativos disponiveis, a performance (do modelo de ML) mantem-se relativamente constante, não havendo um decrescimo de performance por se trabalhar com menos algarismos significativos. Desta forma, podemos cortar casas de algarismos significativos dando mais atenção à ordem de grandeza dos numeros com que se trabalha (cada vez maiores! Basta olhar para o caso de fisica computacional..) visto que a performance é exatamente a mesma.

Resposta Correta: [Nenhuma]