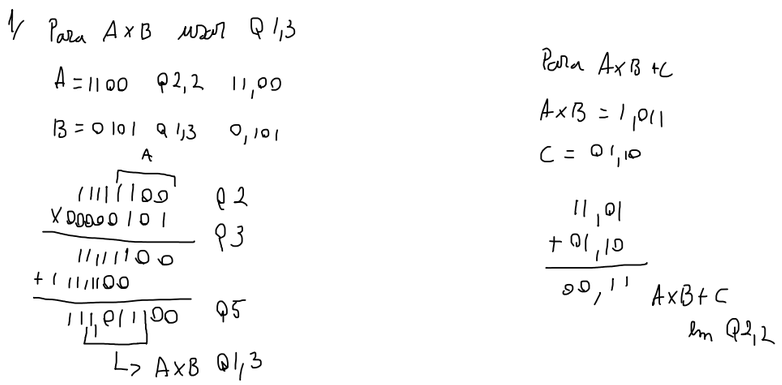
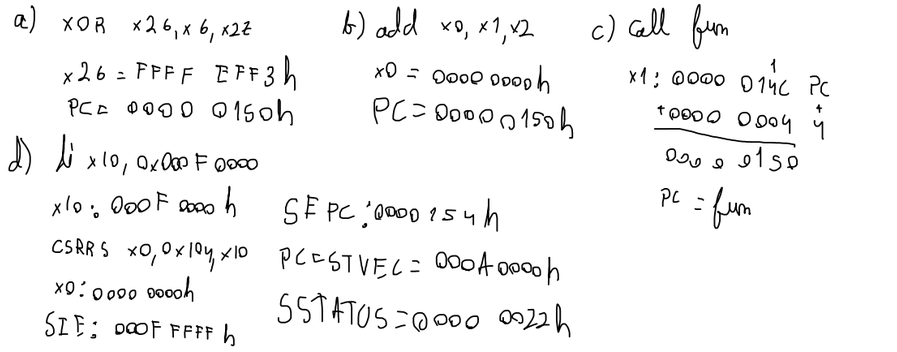
1. Determine algebricamente (em binário) o resultado da operação $A \times B + C$, considerando que todas as operações devem ser realizadas com operandos de 4 bits com sinal. Admita que A=1100 (formato Q2.2), B=0101 (formato Q1.3) e C=0110 (formato Q2.2). Indique qual o formato em virgula-fixa que permite representar o resultado de cada operação com a maior precisão possível. .....[2,0 val.]



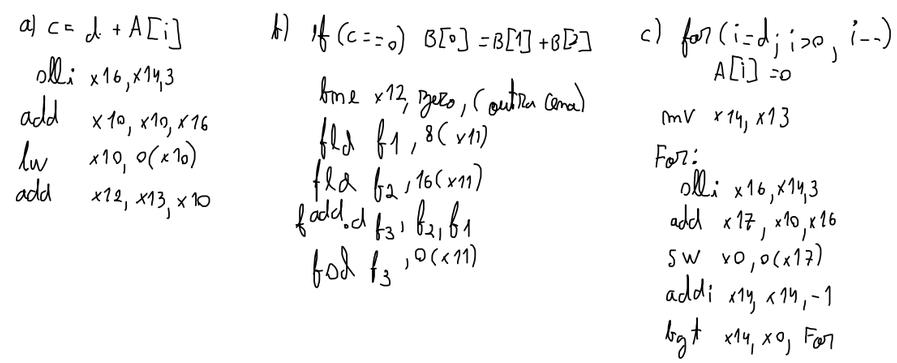
2. Considere o ISA LEGv8 estudado nas aulas teóricas. Para cada uma das seguintes alíneas, indique o número e valor de todos os registos escritos pela execução do troço de código indicado. Utilize o símbolo "?" se não tiver informação suficiente para determinar o seu valor. Considere cada alínea como uma resposta independente. .....[2,0 val.]

X0		**X10**	0x0000 0100	**X20**	0	**R30**	0x0001 021C
X1	0x8000 0000	**X11**	0x0000 0120	**X21**	0	**R31**	0x0000 0000
X2	0x0000 0134	**X12**	0x0000 0140	**X22**	0		
X3	21	**X13**	0x0000 0160	**X23**	0	**PC**	0x0000 014C
X4	37	**X14**	0x0000 0220	**X24**	0		
X5	-3	**X15**	0x0001 0010	**X25**	0	**STVEC**	0x000A 0000
X6	-1	**X16**	0x0002 0FEC	**X26**	0	**SIPC**	
X7	137	**X17**	0x0003 84EF	**X27**	0x0000 100Ch	**SIE**	0x0000 FFFF
X8	15	**X18**	0x0000 FEFF	**X28**	0x0000 120Ch	**SIP**	0x0004 0000
X9	-15	**X19**	0x0000 AABB	**X29**	0x0000 140Ch	**SSTATUS**	0x0000 0002



3. Escreva o código Assembly (usando o número mínimo de instruções) correspondente ao código C indicado em cada uma das alíneas. Para a resolução de cada alínea, admita que x10, x11, x12, x13 e x14 guardam as variáveis *A*, *B*, *c*, *d* e *i*, respetivamente, e ainda que as variáveis foram declaradas da seguinte forma:.....[4,0 val.]

```
long long A[20],c,d,i;
double B[10];
```



4. Considere que pretende traduzir para Assembly do RISC-V o seguinte troço de código C:

```
#define N  20
double VetorA[20];
double VetorB[20];
double VetorC[20];
(…)
for (i=1; i<N; i++) C[i] = calcula(A[i-1],A[i],B[i]);
```

a) Corrija o código Assembly de forma a garantir a funcionalidade indicada, ignorando a convenção do compilador. Preencha na tabela da página seguinte apenas as linhas que devem ser corrigidas. Utilize o código apresentado nesta página como rascunho.....[4,0 val.]

```
1:  .data
2:  VetorA:   .byte 80
3:  VetorB:   .byte 80
4:  VetorC:   .byte 80
(…)
5:              la    x10,VetorA    ; lê o endereço do vetor A
6:              la    x11,VetorB    ; lê o endereço do vetor B
7:              la    x12,VetorC    ; lê o endereço do vetor C
8:  forloop:    addi  x5,x0,20      ; N=20
9:              addi  x6,zero,1     ; i=1
10:             bge   x5,x6,forend  ; while i<N
11:             fld   f0,0(x10)     ; lê A[i-1]
12:             fld   f1,8(x10)     ; lê A[i]
13:             fld   f2,8(x11)     ; lê B[i]
                ; C[i]=calcula(A[i-1],A[i],B[i]);
14:             addi  sp,sp,-24     ; coloca os elementos na pilha por ordem
15:             fsd   f0,16(sp)     ; da chamada à função
16:             fsd   f1,8(sp)
17:             fsd   f2,0(sp)
18:             jal   calcula
19:             fld   f0,0(sp)      ; retira o resultado da função da pilha
20:             addi  sp,sp,8
21:             fsd   f0,0(x12)     ; guarda o resultado em c[i]
                ; atualiza os ponteiros
22:             addi  x10,x10,8
23:             addi  x11,x11,8
24:             addi  x12,x12,8
                ; iteração do for
25:             addi  x6,x6,1
26:             blt   x6,x5,forloop
27: forend:
```

b) Admitindo o seguinte código C relativo à função calcula:

```
double calcula(double A, double B, double C) {
    return (A-B)/C;
}
```

Complete o seguinte troço de código assembly de forma a garantir a correta tradução da função. Ignore as convenções do compilador. .....[3,0 val.]

```
calcula   ; salvaguarda de contexto
          addi sp,sp,-24
          fsd f2,16(sp)
          fsd f1,8(sp)
          fsd f0,0(sp)
          ; leitura dos argumentos da função
          fld     f0, 40(sp)          ; A
          fld     f1, 32(sp)          ; B
          fld     f2, 24(sp)          ; C
          ; calculo do resultado
          fsub.d  f0,f0,f1            ; A-B
          fdiv.d  f0,f0,f2            ; (A-B)/C
          ; retorno do resultado da função
          fsd     f0, 40(sp)
          ; reposição de contexto
          fld f0,0(sp)
          fld f1,8(sp)
          fld f2,16(sp)
          addi sp,sp,40
          ; retorno
          jalr    x0,x1,0
```

5. Responda às seguintes questões tendo em atenção que cada resposta errada corresponde a uma penalização de ¼ da cotação.....[5.0 val.]

a) A Arquitetura do Conjunto de Instruções (em inglês ISA – Instruction Set Architecture), define:
- [x] O funcionamento lógico do processador (instruções, exceções, periféricos).
- [ ] A forma como o processador implementa as instruções.
- [ ] A arquitetura (sistema digital) de suporte à execução das instruções.
- [ ] Nenhuma das anteriores.

b) Indique qual dos seguintes endereços está alinhado a uma palavra de 32-bits:
- [x] AABCh
- [ ] AAB7h
- [ ] DE72h
- [ ] DE7Ah

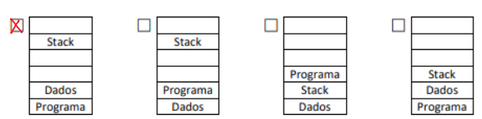
c) Na arquitetura RISC-V, os bancos de registos estão fisicamente localizados:
- [ ] Na memória de instruções
- [ ] Na memória de dados
- [x] No processador, no estágio de Operand Fetch (OF)
- [ ] No processador, no estágio de Instruction Fetch (IF)

d) Qual é a função do PC (Program Counter):
- [ ] Guarda a próxima instrução a executar
- [x] Guarda o endereço da próxima instrução a executar
- [ ] Guarda o número de instruções executadas
- [ ] Guarda o número de instruções do programa

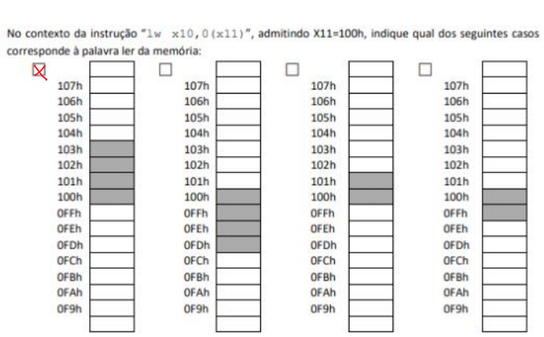
e) A palavra de endereço de acordo com o ISA do RISC-V tem a dimensão de:
- [ ] 8 bits
- [ ] 16 bits
- [x] 32 bits
- [ ] 64 bits

f) A instrução `jal` é fundamental na implementação de:
- [ ] Ciclos (for, while)
- [x] Chamadas a rotinas
- [ ] Retorno de rotinas
- [ ] Tratamento de excepções/interrupções

g) Qual dos seguintes casos corresponde a um mapa de memória típico (os endereços crescem de baixo para cima):



h) No contexto da instrução "`lw x10,0(x11)`", admitindo X11=100h, indique qual dos seguintes casos corresponde à palavra ler da memória:



i) O SP deve ser inicializado:
- [ ] Sempre pelo programador
- [ ] Sempre pelo sistema operativo
- [x] Pelo sistema operativo (se existente) ou pelo programador (caso contrário)
- [ ] Pelo sinal de *reset* do processador

j) O ISA deve definir instruções específicas para acesso aos periféricos:
- [x] Se o método de comunicação com os periféricos for do tipo Port-Mapped I/O
- [ ] Se o método de comunicação com os periféricos for do tipo Memory-Mapped I/O
- [ ] Sempre
- [ ] Nunca

k) Para o tratamento de um interrupção/exceção o processador deve:
- [ ] Esperar que a aplicação termine, antes de executar o código da rotina correspondente
- [x] Esperar que a instrução termine, antes de executar o código da rotina correspondente
- [ ] Abortar a execução da aplicação para executar o código da rotina correspondente
- [ ] Nenhuma das anteriores

l) A chamada à rotina de tratamento de uma interrupção/exceção envolve:
- [ ] Guardar o valor do PC
- [x] Guardar o valor do PC e do GIE
- [ ] Guardar o valor do PC e das SIP
- [ ] Guardar o valor do PC e do SP