Prova com consulta. Duração: 1h30m. Segundo Mini-Teste

Grupo 1. Representação Intermédia de Baixo Nível e Seleção de Instruções (5 valores)
Considere o trecho de código C e a informação relacionada com o tipo e armazenamento das variáveis:

A[i] = B[x] * c; (em que c é uma constante)

Variável	Tipo	Armazenamento (SP indica o registo que armazena o apontador para a pilha)
A	int[]	endereço base na posição 8 de SP
B	int[]	endereço base na posição 4 de SP
i	int	registo r6
x	int	posição 0 a partir de SP

1.a) [2v] Apresente a representação intermédia de baixo nível tendo por base o processador *Jouette* cujas instruções e as respetivas árvores são apresentadas em (*).

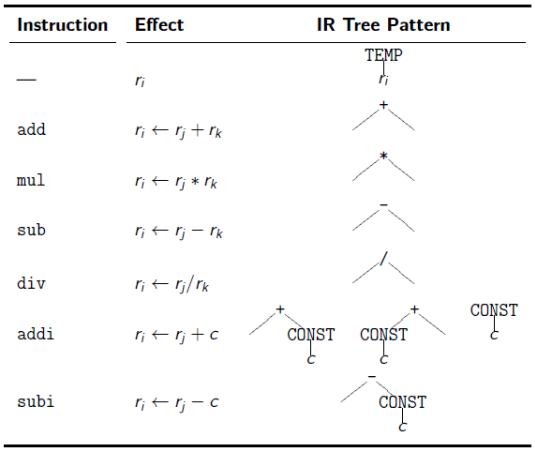
1.b) [3v] Considerando o conjunto de instruções apresentado em (*), utilize o algoritmo *Maximal Munch* para seleccionar as instruções, indicando também a sequência de instruções resultante. Indique na árvore a cobertura da mesma pelas árvores de padrões de instruções seleccionadas.

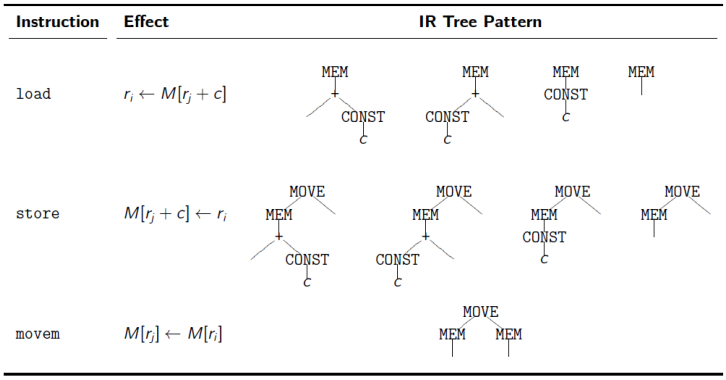
(*) Conjunto de instruções (considere o mesmo custo para todas as instruções) de um processador hipotético designado por *Jouette*:

ADD rd = rs1 + rs2 ADDI rd = rs + c	SUB rd = rs1 − rs2 SUBI rd = rs – c MUL rd = rs1* rs2 DIV rd = rs1/rs2	LOAD rd = M[rs + c] STORE M[rs1 + c] = rs2 MOVEM M[rs1] = M[rs2]

Em que *rd* e *rs* identificam registos de 32 bits do processador (de *r0* a *r31*, tendo *r0* o valor 0), *c* identifica um literal e *M[x]* indica o acesso ao endereço de memória dado por x.

As correspondentes árvores de padrões de instruções são as seguintes:





Grupo 2. Análise de Fluxo de Dados e Alocação de Registos (8 valores)

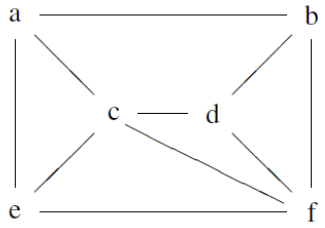
Considere o trecho de código seguinte.

```
1.  int f(int x) {
2.    int z, y, e = 1;
3.    while(x>0) {
4.      z = e*e;
5.      y = e*x;
6.      if(x & 1) {
7.        e = y;
8.      } else {
9.        e = z;
10.     }
11.     x = x - 1;
12.   }
13.   return y;
14. }
```

2.a) [1v] Apresente o grafo de fluxo de controlo (CFG: *Control-Flow Graph*) para o código apresentado.

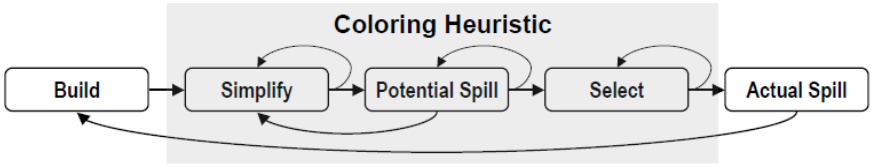
2.b) [2v] Utilize análise de fluxo de dados para determinar o tempo de vida das variáveis e indique o grafo de interferências resultante. Apresente as iterações necessárias para a análise de fluxo de dados apresentando os respetivos conjuntos de *def*, *use*, *in*, e *out*.

Considere o grafo de interferências seguinte.



2.c) [3v] Apresente uma possível alocação de registos a variáveis utilizando o algoritmo de coloração de grafos explicado nas aulas e supondo a utilização de 3 registos (*r1*, *r2*, e *r3*). Apresente o conteúdo da pilha após simplificação do grafo de interferências. No caso de ter de fazer *spilling* indique o critério que usou para a seleção de variáveis e que para este exemplo concreto permita fazer o mínimo número de *spillings*. Apresente o resultado da primeira coloração de grafos (i.e., sem repetir o processo).

2.d) [2v] A figura em baixo apresenta um fluxo possível para a alocação de registos baseada em coloração de grafos. Descreva o que é feito em cada uma das etapas apresentadas e quando é que cada ligação entre etapas é realizada.



Grupo 3. Gerais (7 valores)

Para cada uma das afirmações indique se é verdadeira ou falsa e justifique sucintamente a resposta dada.

3.a) [1,5v] O uso de programação dinâmica para a seleção de instruções pode permitir obter resultados melhores do que os alcançados com o algoritmo *maximal munch*, com a particularidade de explorar mais opções em termos de cobertura da árvore mas evitando a exploração de todo o espaço de soluções.

3.b) [1,5v] Se o CFG (*Control-Flow Graph*) for acíclico é sempre possível determinar o tempo de vida das variáveis com uma iteração (a segunda iteração apenas serviria para verificar que não há alterações nos conjuntos de *in* e de *out* relativamente à primeira iteração).

3.c) [2v] No algoritmo de alocação de registos apresentado nas aulas a ordem com que se colocam os nós do grafo de interferências que obedecem a grau < k (i.e., com número de nós adjacentes menor do que k) na pilha deve ser feita criteriosamente pois isso pode ter impacto no número de *spillings* a efetuar.

3.d) [2v] A solução de Briggs e a solução de George para realizar *coalescing* conduzem na prática a resultados de alocação de registos iguais.

(Fim.)