Universidade do Porto, Faculdade de Engenharia (FEUP)

Mestrado Integrado em Engenharia Informática e Computação

EIC0028-2S – COMPILADORES

Teste - 6 de junho de 2012 *Duração total (Partes I + II): 1 hora e 30 minutos*

Nota: Na parte I as respostas erradas têm cotação negativa. Numa pergunta com 4 alternativas, uma resposta errada tem uma cotação negativa igual a 1/3 da cotação da pergunta. Numa pergunta com 2 alternativas, a resposta errada tem uma cotação negativa igual à cotação da pergunta. As perguntas não respondidas têm cotação 0.

Nome: _____ Número: _____

PARTE I (9 valores)

1. Representação Intermédia de baixo-nível (*low-level intermediate representation*)

1.1 [0.75 val] O principal objetivo da representação intermédia de baixo nível é:

☐ Representar o programa de uma forma mais compacta.

☐ Ter o programa representado por uma lista de instruções.

☐ Fornecer uma representação que facilite as conversões de tipos na análise semântica.

☒ Ter uma representação do programa próxima do código a gerar.

1.2 [0.75 val] A representação intermédia de baixo-nível:

☒ Representa as estruturas da linguagem de programação relacionadas com *loops* e *ifs* por estruturas próximas da forma como *loops* e *ifs* são implementados pela máquina alvo.

☐ É útil para otimizações a nível das chamadas a procedimentos.

☐ Recebe da representação intermédia de alto-nível os resultados da alocação de registos.

☐ É a representação obtida substituindo as variáveis na representação alto-nível por leituras/escritas da/na memória.

1.3 [0.75 val] A representação intermédia de baixo-nível:

☐ É diretamente obtida substituindo os identificadores das variáveis por registos do processador.

☐ É obtida a partir das instruções geradas para a máquina alvo.

☐ É diretamente obtida substituindo cada chamada a um procedimento pelo código do procedimento.

☒ Nenhuma das outras alíneas está correta.

2. Análise do Fluxo de Dados

2.1 [0.75 val] A análise do fluxo de dados determina os tempos de vida das variáveis:

☐ Percorrendo a representação intermédia de baixo nível uma vez e inspecionando as definições das variáveis em cada instrução.

☐ Verificando se as variáveis são inicializadas e se são devolvidas pelas funções.

☐ Por inspeção do grafo de interferências entre variáveis.

☒ Todas as outras alíneas estão incorretas.

2.2 [0.75 val] A análise do fluxo de dados:

☐ É uma técnica usada na seleção de instruções.

☒ É uma técnica que permite fazer a análise da vitalidade das variáveis.

☐ Permite verificar se nas representações intermédias os dados fluem de acordo com o programa.

☐ Todas as outras alíneas estão incorretas.

2.3 [0.75 val] Para determinar o tempo de vida das variáveis:

☐ Nunca são necessárias várias iterações na análise do fluxo de dados.

☐ É necessário criar o grafo de interferências entre variáveis.

☐ É mais eficiente se a análise do fluxo de dados for feita pela ordem direta do fluxo de instruções.

☒ Todas as outras alíneas estão incorretas.

3. Alocação de Registos

3.1 [0.75 val] O grafo de interferências entre registos/variáveis:

☐ Pode ser obtido por análise direta de cada instrução no código.

☐ Ilustra fundamentalmente as incompatibilidades entre registos/variáveis.

☒ É construído depois da análise do tempo de vida dos registos/variáveis.

☐ É obtido diretamente da tabela de símbolos.

3.2 [0.75 val] Indique a opção correta:

☐ Nunca precisamos de fazer *register spilling* pois podemos considerar que o processador tem um número muito elevado de registos disponíveis.

☐ Quando marcamos uma variável para *register spilling* nunca temos de voltar a fazer a alocação de registos.

☒ Caso tenhamos de utilizar registos auxiliares nas instruções de *register spilling*, depois de identificarmos as variáveis que necessitam de *spilling* e de realizarmos a alocação de registos para as outras variáveis, temos de voltar a realizar a análise do tempo de vida.

☐ Na prática um compilador nunca precisa de fazer *register spilling* pois pode considerar que os valores de todas as variáveis são armazenados em memória.

3.3 [0,75 val] A utilização de *register coalescing*:

☐ É apenas uma forma de reduzir o número de nós no grafo de interferências.

☐ Não traz grandes vantagem pois a própria coloração de grafos pode decidir de qualquer modo se atribui o mesmo registo para as duas variáveis ou não.

☒ Pode originar dificuldades na coloração do grafo de interferências.

☐ É apenas uma forma de aumentar o número de vizinhos de um nó no grafo de interferências.

4. [2,25 val] Indique se cada uma das seguintes afirmações é verdadeira ou falsa:

V	F	
☐	☒	Não é sempre possível evitar iterações da coloração de grafos quando se tem de realizar *register spilling*, mesmo que para isso o compilador afete um registo do processador para lidar unicamente com os *load/store* necessários para implementar o *spilling*.
☒	☐	A seleção de instruções pode ser feita com o uso de templates (esqueletos) e de algoritmos de cobertura sobre árvores ou grafos que representam as expressões.
☐	☒	A representação intermédia de alto-nível deve ser dependente da linguagem e da máquina alvo.
☐	☒	O problema de seleção de instruções é resolvido por programação dinâmica devido à menor complexidade computacional da programação dinâmica face ao algoritmo *Maximal Munch*.
☐	☒	Se escolhermos uma sequência de instruções adequada para realizar a análise do fluxo de vida das variáveis o número máximo de iterações necessárias pelo algoritmo de fluxo de dados é sempre dois.
☒	☐	A utilização de *register coalescing* não é sempre vantajosa na alocação de registos.
☐	☒	Na prática quando se tem de escolher uma variável para *spilling* essa escolha pode ser aleatória pois o resultado final vai ser o mesmo.
☐	☒	O cálculo do tempo de vida das variáveis é feito por um algoritmo iterativo cujos resultados finais variam com a ordem com que as instruções de uma função são analisadas.
☒	☐	O algoritmo *Maximal Munch* não produz sempre a seleção de instruções ótima (de menor custo).
☐	☒	Os compiladores não respeitam algumas das interferências no grafo de interferências pois duas ou mais variáveis podem ser empacotadas em apenas um registo do processador.