Segunda Prova de Análise e otimização de Código
- DCC888 -
Ciência da Computação

Nome: _____
"Eu dou minha palavra de honra que não trapacearei neste exame."

Número de matrícula: _____

As regras do jogo:

- A prova é sem consulta.
- Quando terminar, não entregue nada além do caderno de provas para o instrutor.
- Cada estudante tem direito a fazer uma pergunta ao instrutor durante a prova. Traga o caderno de provas quando vier à mesa do instrutor.
- A prova termina uma hora e quarenta minutos após seu início.
- Seja honesto e lembre-se: **você deu sua palavra de honra**.

Alguns conselhos:

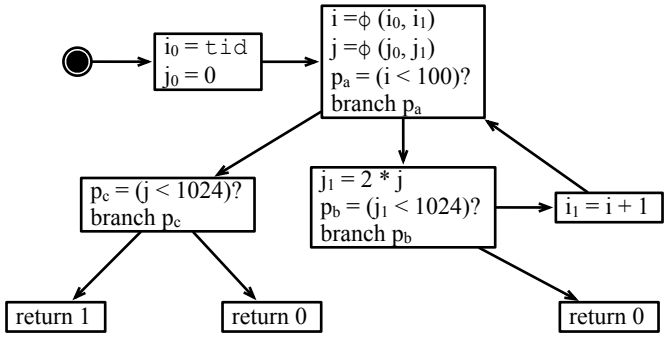
- Escreva sempre algo nas questões, a fim de ganhar algum crédito parcial.
- Você pode sacrificar sua pergunta para saber por que Hilda deixou a vida de princesa para entrar na vida de rameira, na humilde opnião desse instrutor.
- Se não entender alguma questão, e já tiver gasto sua pergunta, escreva a sua interpretação da questão junto à resposta.
- A prova não é difícil, ela é divertida, então aproveite!

Tabela 1: Pontos acumulados (para uso do instrutor)

Questão 1	Questão 2	Questão 3	Questão 4

Ponto Extra: cite um dos 12 trabalhos que Hércules realizou a mando de Euristeu, rei de Tirinto.

1. Esta questão refere-se ao programa abaixo. Esse programa será executado em um sistema de processamento paralelo do tipo *Single Instruction, Multiple Data*. Nesse ambiente, temos várias *threads*, todas executando a mesma instrução, porém sobre dados diferentes. Cada *thread* possui um identificador `tid`, o qual é diferente para cada *thread*.



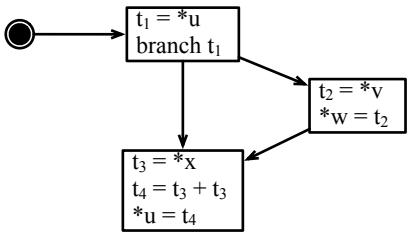
(a) (2 Pontos) Uma variável é dita *uniforme* quando seu valor é o mesmo para todas as *threads* que executam em uma máquina SIMD. o programa acima possui uma variável que é tanto uniforme quanto divergente. Que variável é essa?

(b) (4 Pontos) É possível dividirmos a linha de vida da variável da questão anterior para que todas as variáveis do programa fossem, ao longo de suas linhas de vida, totalmente divergentes, ou totalmente uniformes. Redesenhe o programa, com as linhas de vida já divididas. Escreva o predicado que controla cada função ϕ ao lado daquela função.

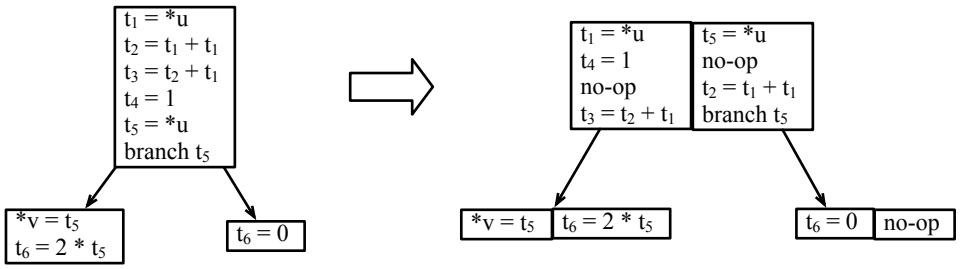
(c) (1 Ponto) Quais dentre as variáveis do programa produzido na questão anterior são uniformes?

(d) (3 Pontos) Um desvio condicional, isto é, uma instrução do tipo `branch`, pode ser *divergente* ou *uniforme*. Um desvio é divergente se ele pode ser percorrido de forma diferente por *threads* distintas. Quais dentre os desvios do programa acima são divergentes?

2. (10 Pontos) Esta questão refere-se ao programa abaixo, o qual será executado em uma arquitetura especial, que possui dois buscadores de instrução, e duas unidades de processamento. Em outras palavras, essa nossa arquitetura consegue buscar duas instruções ao mesmo tempo, desde que elas tenham sido emparelhadas pelo compilador.

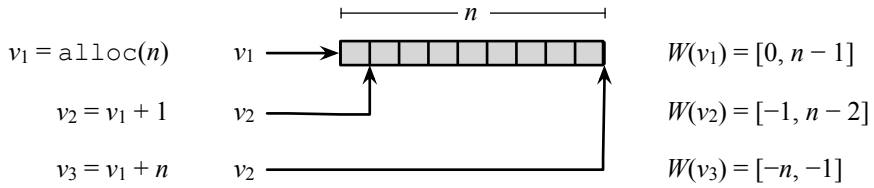


Nessa arquitetura, uma instrução de carregamento como `t = *v` demora dois ciclos para terminar. Assim, a primeira sequência de instruções: `t1 = *v; branch t1`, na verdade, vai executar em três ciclos, isto é: `t1 = *v; no-op; branch t1`. Nesta questão, pede-se que você modifique o programa, **sem alterar sua semântica**, de modo a diminuir o tempo de espera entre as instruções, e permitir o seu processamento paralelo. Um exemplo do que é pedido está mostrado logo a seguir:



Lembre-se que você pode replicar instruções, desde que isso não altere a semântica do programa.

3. Nesta questão você deve criar uma análise que infira o tamanho de regiões alocadas dinamicamente. A sua análise deve associar a cada ponteiro um intervalo W de *off-sets* válidos, conforme descrito na figura abaixo:



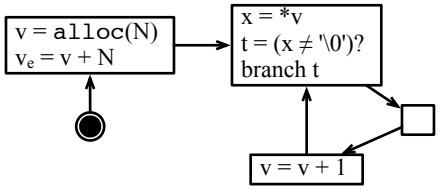
Se v é um ponteiro, e $W[v] = [l, u]$, então sabe-se que todo o espaço entre $v + l$ e $v + u$ são regiões alocadas de memória. Os limites l e u podem ser números inteiros, nomes de variáveis, ou os valores especiais $-\infty$ e ∞. Você pode assumir a existência de uma análise de largura de variáveis R, que associe a cada variável inteira v (que não seja um ponteiro), um interval $[a, b]$.

(a) (5 Pontos) Assuma a existência das seguintes instruções:

- $v = N$; inicialização de uma variável com valor desconhecido N. Assuma que $R(N) = [N, N]$ é um intervalo simbólico.
- $v = c$, $c \in N$; atribuição de constante à variável.
- $v_1 = v_2$; atribuição entre variáveis.
- $v_1 = \texttt{alloc}(v_2)$; alocação de v_2 espaços de memória ao ponteiro v_1.
- $\texttt{free}(v)$; liberação da memória apontada por v.
- $v_1 = v_2 + c$, v_2; aritmética de ponteiros: soma da constante c ao ponteiro v_2.

Sua análise deve ser **densa**, isto é, você precisa associar pares, variáveis e pontos de programa à informação. Nesse caso, a informação são intervalos de *off-sets* válidos.

(b) (5 Pontos) Uma questão importante no projeto dessa análise é garantir a sua terminação. Defina uma forma de **alargamento** para assegurar que a sua análise termine. O alargamento é tão somente uma maneira de garantir a terminação da análise. Mostre o resultado de sua análise quando aplicada ao programa abaixo:



4. (10 Pontos) Esta questão refere-se às regras de avaliação mostradas logo abaixo. Essas regras descrevem a semântica de passo longo de uma linguagem que possui expressões condicionais e negação:

$$\frac{t_1 \Downarrow \text{true} \quad t_2 \Downarrow v}{\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \Downarrow v} \quad \text{[B-TRUE]}$$

$$\frac{t_1 \Downarrow \text{false} \quad t_3 \Downarrow v}{\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \Downarrow v} \quad \text{[B-FALSE]}$$

$$\frac{t \Downarrow \text{true}}{\text{not } t \Downarrow \text{false}} \quad \text{[B-NOTTRUE]}$$

$$\frac{t \Downarrow \text{false}}{\text{not } t \Downarrow \text{true}} \quad \text{[B-NOTFALSE]}$$

Prove, usando as regras acima, a seguinte equivalência: `if t1 then t2 else t3` $\equiv$ `if not t1 then t3 else t2`