Verificando descontos para cafés!

EP3 - Organização de Computadores

Entrega: 14/12/2008

1 Introdução

Após a conturbada entrega do EP2, recebemos diversas reclamações por parte dos computadores da rede linux. De forma geral, a reclamação pode ser resumida para:

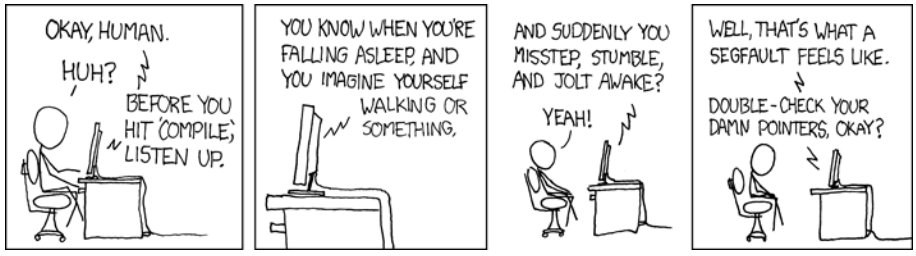

Figura 1: XKCD 371 - http://xkcd.com/371/

Para tentar resolver esse problema, achamos que café é a solução! Por isso queremos fazer o pedido de uma máquina de café para a rede linux. Acontece que alguns alunos precisam de mais café do que outros e não queremos arruiná-los. A solução encontrada para isso foi dar descontos (5%, 10% ou 20%) de acordo com a necessidade de cada grupo de alunos.

Instalar botões com o valor do desconto na máquina obviamente não é uma solução. Por tanto decidimos que cada aluno poderá pedir um cartão magnético que conterá um identificador para o aluno de 20 bits. Queremos que vocês implementem um circuito que realiza algumas operações nesse bits para descobrir que eles são válidos (algo no estilo MD5 sum[1] mas que vocês inventarem).

[1] `http://en.wikipedia.org/wiki/Md5sum`

2 Definição do trabalho

A definição desse circuito deve ser feita usando VHDL. Vocês devem lidar diretamente com os bits e só podem usar operadores lógicos. Vocês devem testar seu circuito com entradas válidas e inválidas.

Seu algoritmo deve permitir que o número de alunos sem desconto seja maior que o de alunos com desconto de 5% que seja maior que os de 10% que seja maior que os de 20%. O algoritmo também será avaliado com relação à facilidade de gerar um identificador aleatoriamente e ele ser válido para descontos de 5%, 10% e 20%.

Para escrever e testar seu código, vocês podem usar o GHDL[2]. Para instalá-lo, é necessário ter direitos de administrador. Se vocês não têm acesso a um computador no qual têm direitos de administrador, estamos pedindo que o GHDL seja instalado em alguns computadores da rede Linux. Favor se informar junto ao professor ou aos administradores da rede Linux.

Segue uma descrição de como instalar num Ubuntu 8.10 novo.

```
usuario@maquina$ su
Senha:
root@maquina# apt-get install build-essential zlib1g-dev
...
root@maquina# curl http://ghdl.free.fr/ghdl-0.27.tar.bz2
...
root@maquina# tar -C / -jxvf ghdl-0.27.tar.bz2
...
root@maquina# exit
exit
usuario@maquina$ $ghdl -v
GHDL 0.27 (20080701) [Sokcho edition]
...
```

3 Entrada

Seu verificador deve receber 20 bits do cartão magnético.

Do lado dos testes, ambos arquivos de teste devem executar sem entrada. O arquivo `verificador_test.vhdl` deve conter várias entradas válidas. Pelo menos duas entradas de cada valor de desconto válido e três exemplos de valores inválidos que dão um sinal de erro (ou desconto 0).

[2] http://ghdl.free.fr/

4 Saída

Seu processo de verificação deve enviar um sinal em d0 caso o identificador do aluno não seja válido. Caso o identificador seja válido, o verificador deve enviar um sinal em d5, d10 ou d20 de acordo com o desconto que o aluno deve receber.

Já para os testes, o arquivo `verificador_test.vhdl` não deve imprimir nada. Irei modificar bits aleatórios da matriz de exemplo para garantir que os exemplos nele, de fato, imprimem algo quando inputs errados são testados.

5 Formato de entrega

Você deve entregar sua solução usando o sistema PACA disponível em:

`http://paca.ime.usp.br`

Sua solução deve consistir de:

1. Código fonte do programa

   Você e sua equipe deverão empacotar todos os arquivos necessários para que eu compile e rode sua solução. Tipicamente isso incluirá os seguintes arquivos:

   `Makefile`, `src/verificador.vhdl`, `test/verificador_test.vhdl`, `relatorio.pdf`

   Se vocês incluirem mais arquivos, tenham certeza de adicioná-los ao pacote. Por favor, NÃO incluam os binários gerados ou arquivos temporários (como arquivos .o). Eu devo conseguir rodar `make` para compilar e rodar o seu programa.

2. Um relatório do trabalho em PDF

   Desta vez, quero uma descrição da idéia de validação dos seus 20 bits além do algoritmo em pseudo-código mais alto nível que vocês usaram para fazer a verificação. Sua nota estará diretamente relacionada com a facilidade que eu tiver de entender seu algoritmo então expliquem bem. Também quero saber quantos estudantes válidos para cada tipo de desconto o algoritmo de vocês permite e mais dois possíveis algoritmos bem diferentes do que vocês implementaram. Um deles deve permitir mais estudantes e o outro menos. Como sempre, quero uma avaliação da linguagem e do impacto de se trabalhar num nível muito mais baixo para o desenvolvimento de algoritmos.

3. **(Opcional)** Um gerador de identificadores

   Como bônus, quero o código fonte de um programa C que gera todas as combinações válidas para cada tipo de desconto de forma ordenada e dividida. Este programa deve permitir que seja fácil achar um identificador dado um tipo de desconto. A compilação e execução desse programa deve ser possível com `make gerador` e `./gerador` respectivamente.

A entrega deve consistir de um único arquivo em formato tar.gz contendo todos os arquivos citados acima. O nome desse arquivo deve seguir o seguinte formato:

nome-integrante1-nome-integrante2-ep3.tar.gz

e ele deve se extrair em uma pasta com o mesmo nome (sem a extensão) com todos esses arquivos. Tomem cuidado para entregar esse arquivo apenas uma vez para cada grupo. Apenas um membro de cada equipe deve entregar o arquivo. Caso contrário, pegarei o arquivo entregue mais tarde (e descontarei nota). Para cada item fora do formato de entrega pedido, o grupo perderá um ponto na nota final.

`0x446976697274616d2d736521`