INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

Departamento de Eletrônica
Prof. Charles Borges de Lima.

Programando um Switch/Case

Objetivo

Estudar uma alternativa eficiente para a geração de código associado à construção `switch/case`, a qual é suportada por linguagens de alto nível como opção ao aninhamento de desvios condicionais `if-then-else`. Essa alternativa é baseada no uso de uma tabela de endereços de desvio (JAT – *jump address table*).

Estudo de Caso

No código abaixo (escrito em linguagem C), uma, e somente uma, dentre quatro alternativas (mais o caso *default*) de cálculo do valor de `f` é selecionada para ser executada dependendo do valor da variável `k`, onde `k` é um inteiro no intervalo $0 \leq k \leq 3$. A semântica do código pressupõe que para valores de `k` fora desse intervalo o caso `default` deve ser executado. Além disso, a inserção de comandos `break` após o cálculo de `f` denota que as alternativas são mutuamente exclusivas.

```
switch(k)
{
   case 0:   f  = i + j;      break;    // k = 0
   case 1:   f  = g + h + k;  break;    // k = 1
   case 2:   f  = g - h;      break;    // k = 2
   case 3:   f  = i - j;      break;    // k = 3
   default:  f  = i - j + h;            // k  fora do intervalo [0,3]
}
```

Convenções para o exercício

- Adote a seguinte alocação de registradores:
  (f, g, h, i, j, k) → ($s0, $s1, $s2, $s3, $s4, $s5)
- Atribua rótulos `L0`, `L1`, `L2`, `L3` e `default` às posições de memória onde são iniciadas as instruções que codificam cada um dos casos de execução (`k=0`, `k=1`, `k=2`, `k=3` e `k` fora do intervalo [0,3], respectivamente).
- Atribua o rótulo `exit` à posição de memória que representa a primeira instrução após a execução do trecho do programa.

Procedimento de teste

O procedimento de teste usa a seguinte inicialização de referência para todas as variáveis (exceto `k`): `f = 0`, `g = 4`, `h = 1`, `i = 4` e `j = 6`. Essa inicialização é fixada na área de dados globais do programa para facilitar o teste de forma que apenas o valor de `k` é alterado para cada teste.

1. Atribua sucessivamente os valores (-1, 0, 1, 2, 3, 4) à variável k e monitore o conteúdo do registrador $s0. Não se esqueça de montar novamente o programa cada vez que um valor diferente for atribuído a variável k.

2. Verifique se os resultados esperados (-1, 10, 6, 3, -2, -1) são obtidos em $s0.

Exercício 1

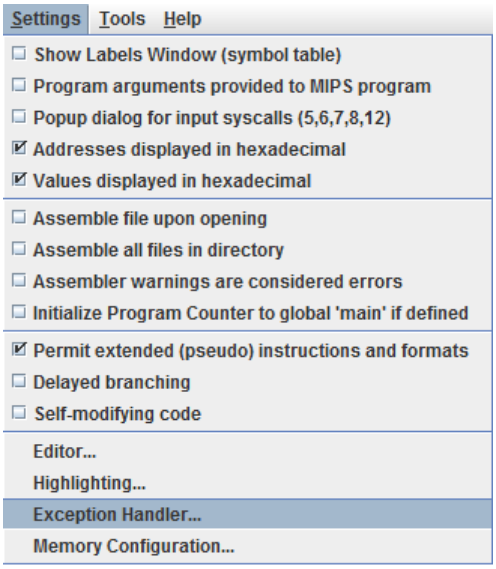
a) Conforme estrutura do arquivo de programa abaixo, usando uma JAT, programe em linguagem de montagem do processador MIPS, o trecho de código do estudo de caso.

```
.data
# Seção 1: variáveis f, g, h, i, j armazenadas em memória (inicialização)
_f: .word 0
_g: .word 4
_h: .word 1
_i: .word 4
_j: .word 6

# Seção 2: jump address table
jat:
.word L0
.word L1
.word L2
.word L3
.word default

.text
.globl main
main:
# Seção 3: registradores recebem valores inicializados (exceto variável k)
lw $s0, _f
lw $s1, _g
lw $s2, _h
lw $s3, _i
lw $s4, _j
la $t4, jat    #carrega endereço base de jat

# Seção 4: testa se k esta no intervalo [0,3], caso contrário default
...
# Seção 5: calcula o endereço de jat [k]
...
# Seção 6: desvia para o endereço em jat[k]
...
# Seção 7: codifica as alternativas de execução
...
exit:
```

- **Seção 1**: estabelece os valores das variáveis de interesse (exceto `k`) que devem residir inicialmente na memória.
- **Seção 2**: descreve o conteúdo de JAT (os endereços das alternativas de execução são aqui representados pelos rótulos `L0-L3` e `default`).
- **Seção 3**: descreve a inicialização dos registradores alocados para as variáveis (exceto `k`). Ao final dessa seção, a pseudo-instrução `la $t4,jat` carrega em `$t4` o endereço base de JAT.
- **Seção 4**: nesta seção escreva o trecho de código usando apenas duas instruções nativas para testar se `k` está no intervalo apropriado e desviar para o caso default quando k esteja fora de tal intervalo.
- **Seção 5**: nesta seção, calcule o endereço efetivo de JAT[k]. Lembre-se que cada elemento da JAT é uma palavra que representa um endereço de memória. Além disso, multiplique o valor de k por 4 para obter o deslocamento em bytes em relação ao endereço-base (início) da JAT.
- **Seção 6**: primeiramente, carregue o conteúdo de JAT[k] em um registrador temporário `$t0`. Em seguida, programe um desvio para o endereço armazenado em `$t0`.
- **Seção 7**: codifique as cinco alternativas de execução identificadas pelos rótulos `L0`, `L1`, `L2`, `L3` e `default`.

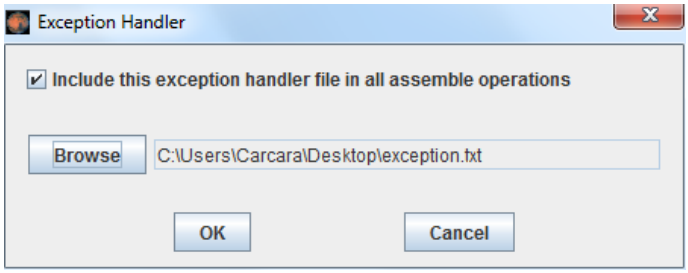
Restrições: nas seções de 4 a 7 você deve usar somente instruções nativas. Para soma e subtrações use instruções `add` e `sub` (as quais detectam *overflow*). Não armazene o valor `k` na memória.

b) Configure o simulador conforme figura abaixo (opção Settings).



c) Crie um arquivo vazio **exception.txt** e salve no Desktop (por exemplo, usando o programa Notepad).

d) Atribua o arquivo **exception.txt** para o manipulador de exceções, conforme apresentado abaixo (opção Settings → Exception Handler).



e) Salve o programa (opção File → Save as).

f) Simule a execução do código (teclas F5 e F7) e verifique seu funcionamento de acordo com o procedimento de teste definido acima. Adapte o código até que os resultados esperados sejam alcançados.

1 – Qual o endereço de memória do primeiro elemento armazenado na JAT?

```
jat: 0x_____________
```

2 – A que endereços efetivos de memória correspondem os seguintes rótulos?

```
L0: 0x_____________   L1: 0x_____________   L2: 0x_____________

L3: 0x_____________   default: 0x_____________
```

3 – A pseudo-instrução `la $t4,jat` foi expandida pelo montador em termos de duas instruções nativas no MIPS. Reproduza-as abaixo em linguagem de montagem usando os nomes simbólicos dos registradores utilizados no código. Indique também o respectivo código em linguagem de máquina, representado em hexadecimal.

Linguagem de montagem	**Linguagem de máquina**

4 – As instruções que carregam os valores inicializados de `f`, `g`, `h`, `i` e `j` em registradores não fazem parte do código gerado para o `switch`, mas apenas servem para evitar que você tenha que inicializá-las cada vez que mudar o valor de `k`. Entretanto, é interessante notar que a carga de cada um daqueles valores usa uma pseudo-instrução, que é expandida em duas instruções nativas. Mostre a expansão para a pseudo-instrução `lw $s0, _f` (em linguagem de montagem), usando os nomes simbólicos dos registradores utilizados no código.

Linguagem de montagem

5 – Para o código da questão 4, qual o valor atribuído a `$at`?

`$at: 0x_____`

6 – Mostre o código em linguagem de montagem que testa se `k` pertence a `[0,3]` (seção 4) e desvia para o endereço `default`.

_____

_____

Dica: para responder às próximas questões, compare as informações das colunas Code e Basic da janela de código.

7 – Qual a codificação em linguagem de máquina da instrução `j exit`?

`j exit` ⇔ `0x_____`

8 – O rótulo `exit` foi resolvido pelo montador. Que endereço lhe foi atribuído?

`exit` ⇔ `0x_____`

9 – Qual o valor dos 26 bits menos significativos da representação binária da instrução `j exit` ?

25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

10 – Comente a consistência dos valores obtidos nas questões 8 e 9, mostrando como um deles é obtido a partir do outro.