UFPR – Departamento de Informática – BCC

CI212 – Organização e Arquitetura de Computadores

Exercícios sobre Tratamento de Exceções de Memória Virtual

1) Considerando que o código abaixo descreve as estruturas de dados necessárias, escreva, em C, uma função para o tratador de faltas na TLB de dados. Suponha que os registradores da interface da TLB estão mapeados no endereço físico `0xFFFF.0000`, e que a estrutura `TP` é alocada no endereço `0xF000.0000`. Se as estruturas, como definidas, não são suficientes, acrescente os componentes necessários e justifique os acréscimos. Explique claramente suas suposições quanto a quaisquer estruturas em hardware ou em software de que seu código depende.

```
typedef unsigned int uint;
typedef struct {
   uint:1  val; // valido
   uint:1  usd; // usado
   uint:1  mod; // modificado
   uint:1  wr;  // writable
   uint:1  ex;  // executable
   uint:20 pfn; // num pag fisica
   uint:7  pad; // enchimento
} elemTP;       // sizeof(elemTP)=4
elemTP TP[2**20]; // tab de paginas
```

```
typedef struct {
   elemTP map[32]; // 32 elementos
   uint:5  rand;   // vitima aleatoria
   uint:1  pgFlt;  // falta de pagina
   uint:26 pad;    // enchimento
   uint epc;       // NOVO: errorPC
   uint etq[32];   // NOVO: etiquetas
} TLB;

TLB dtlb;          // TLB
TLB *pTLB;  elemTP *pTP;
```

`elemTP` tem 32 bits de largura, com 7 bits de *padding* (enchimento) para completar a palavra de 32 bits.

campo	val	usd	mod	wr	ex	pfn	pad
# bits	1	1	1	1	1	20	7

`TLB` tem duas matrizes de 32 bits de largura (`map` e `etq`), 6 bits de status, 26 bits de *padding* e 32 bits para `EPC`.

campo	map[32]	rand	pgFlt	pad	epc	etq[32]
# bits	32x32	5	1	26	32	32x32

Da maneira como as estruturas de dados estão, a TLB é totalmente associativa com reposição aleatória (`dtlb.rand`). Note que o tratamento em caso de acerto na TLB é irrelevante nesta questão, e que a pergunta pressupõe conhecimento de *bit fields* em C. Um tratador de exceção é invocado pela CPU diretamente de um vetor de tratadores (vetor de interrupções) e portanto *deve* ser código rápido e simples –não recebe parâmetros e não invoca outras funções desnecessariamente.

A resposta solicitada é o código para o tratador da exceção de *falta na TLB*. Isso significa que o registrador EPC (errorPC) contém o endereço em que ocorreu a falta porque este endereço é armazenado no EPC como parte do tratamento da exceção pela CPU. Ao menos são duas as alternativas: (i) a exceção grava o EPC num lugar bem-conhecido —um registrador na TLB, ou (ii) o tratador recupera este endereço da pilha. A resposta que segue pressupõe que o EPC é armazenado na TLB. Serão usados 2 apontadores, um para a TP (`pTP`) e outro para a TLB (`pTLB`).

Para que seja possível copiar os bits de status do registro na TLB que será sobre-escrito (a vítima da reposição) para a tabela de páginas é necessário armazenar o número da página virtual (nPV) na TLB (como etiqueta) para que se possa indexar a TP com o nPV. Deve-se incluir nas estruturas acima um vetor de páginas virtuais na TLB de forma que `dtlb.etq[i]` contenha o número da página virtual correspondente à página física armazenada em `dtlb.map[i]`.

No MIPS uma falta de página só é descoberta depois de uma falta na TLB —o tratador da falta na TLB descobre a falta de página e marca sua ocorrência no registro da TLB (`dtlb.pgFlt=1`) e a instrução é buscada novamente. Quando a segunda exceção é detectada inicia-se então o tratamento da falta de página com outra rotina de tratamento.

```
// codigo de inicializacao
pTLB = 0xFFFF0000;   // endereco da TLB em variavel global
pTP  = 0xF0000000;   // endereco da tabela de paginas em variavel global
...
void tratadorFaltaNaTLB(void) {
   uint i = (uint)pTLB->rand;            // vitima escolhida aleatoriamente
   elemTP vitima = pTLB->map[i];         // salva informacao da vitima
   elemTP *endVitima, *endFalta;
   if ( pTLB->pgFlt ) {                  // e' falta de pagina???
      // trata FALTA DE PAGINA ou SEGMENTATION FAULT -- nao nesta questao
   } else {                              // nao e' falta de pagina
      if ( vitima.val ) {                   // salva informacao da vitima na TP
         endVitima = (pTP + pTLB->etq[i]);  // indice da vitima na TP
         endVitima->usd = vitima.usd;       // foi usada
         endVitima->mod = vitima.mod;       // foi modificada
      } // fim da informacao da vitima

      endFalta = pTP + ((pTLB->epc)>>12)*4;  // numero da pagina faltante

      if ( endFalta->val ) {             // pagina mapeada em memoria
         vitima = *endFalta;                // copia valores da TP para TLB
         vitima.usd = 1;                    // foi usado agora
         vitima.val = 1;                    // esta' valido
         pTLB->map[i] = vitima;             // atualiza TLB com mapa EV→EF
         pTLB->etq[i] = (pTLB->epc)>>12;    // preenche etiqueta
      } else {                           // pagina em disco/swap -- page fault
         pTLB->map[i].val = 0;              // marca mapeamento como invalido e...
         pTLB->pgFlt = 1;                   // avisa da falta de pagina
         // provocara' nova excecao, desta vez como FALTA DE PAGINA
      } // pagina em disco
   } // nao e' falta de pagina
}
```

2) Considerando que o código abaixo descreve as estruturas de dados necessárias, escreva, em C, uma função para o *tratador de faltas de página*. Suponha que os registradores da interface da TLB estão mapeados no endereço físico `0xFFFF.0000`, e que a estrutura `TP` é alocada no endereço `0xF000.0000`. Se as estruturas, como definidas, não são suficientes, acrescente os componentes necessários e justifique os acréscimos. Explique claramente suas suposições quanto a quaisquer estruturas em hardware ou em software de que seu código depende. A função `encontraPaginaLivre()` retorna o número de uma página física livre (quadro disponível), e a função `lePaginaDoDisco()` retorna `TRUE` se a página solicitada foi lida com sucesso da área de *swap* e copiada do disco para a memória, no local indicado. Considere qye estas duas funções já estão disponíveis e são corretas.

```
typedef unsigned int uint;
typedef struct {
   uint:1  val;
   uint:1  usd;
   uint:1  mod;
   uint:1  wr;
   uint:1  ex;
   uint:20 npf
   uint:7  pad;
} elmtTP;
```

```
uint encontraPaginaLivre(void);
typedef struct {
   elemTP  map[32];
   uint:5  rand;
   uint:1  pgFlt;
   uint:26 pad;
   uint    epc;
   uint    etq[32];
} TLB;
```

```
uint lePaginaDoDisco(uint npf);


elemTP TP[2**20];
TLB dtlb;
```

3) Ajuste sua resposta para a questão acima supondo que a tabela de páginas é hierárquica, em dois níveis.