Questionário do Trabalho 2 - Arquitetura de Computadores II

André Felipe Romero Faccio, Kellven Dias dos Santos Rodrigues

1 Comandos de compilação usados

```
# python simpred arquivoTrace nLinhasBPB
# Por exemplo:

python simpred.py trace_rijndael_encoder_mi.txt 16
```

2 Experimento 1: Indexação do BPB

(a) Para as técnicas de predição 1-bit e predição 2-bits, o número de linhas do BPB (nLinhasBPB) é sempre uma potência de 2. Por quê?

O número de linhas do Branch Prediction Buffer (BPB) é sempre potência de 2 por razões de eficiência na indexação. O acesso ao BPB é feito por meio do endereço da instrução de desvio. Para localizar rapidademnte a entrada correspondente no buffer, utiliza-se parte dos bits menos significativos do endereço da instrução para formar um índice. Portanto, com uma potência de dois, esse índice pode ser facilmente obtido através de operações binárias.

(b) Os traces fornecidos foram gerados executando os programas em um simulador de um processador com as seguintes características:

- Todas as instruções têm tamanho de 32 bits.
- O processador trabalha com palavras de 32 bits.
- A memória principal é endereçada por bytes.
- Os endereços de memória são de 32 bits.

Para as técnicas de predição o 1-bit e 2-bits, o BPB é indexado usando parte (alguns bits) do endereço da instrução de desvio condicional. O número de bits usados é $\log_2(nLinhasBPB)$. Os bits usados são os bits menos significativos do endereço, excluindo-se os bits de posição 0 e 1.

(b.1) Por que os dois bits menos significativos do endereço da instrução de branch não são usados para indexar o BPB?

Neste processador, todas as instruções têm tamanho fixo de 32 bits (4 bytes) e a memória é endereçada por bytes. Logo, os endereços de instruções sempre serão múltiplos de 4, isto é, os dois bits menos significativos sempre serão 00. Portanto, usá-los no índice do BPB seria inútil, pois não acrescentariam nenhuma informação e levaria a várias instruções diferentes mapeadas para a mesma entrada do BPB.

(b.2) O que aconteceria se os bits $\log_2(nLinhasBPB)$ bits usados iniciassem na posição 0?

Se isso acontecesse, então os bits fixos 00 fariam com que vários endereços diferentes mapeassem para os mesmos índices do BPB, causando vários conflitos, redução da acurácia de predição e o desempenho caíria drasticamente.

3 Experimento 2: Técnicas Estáticas

(a) Para cada técnica de predição estática, calcule:

- A taxa de acertos para cada trace indicado
- A taxa de acertos média (entre todos os traces) (média aritmética simples)

As taxas de acerto devem ser apresentadas na forma de porcentagem, arredondadas para 2 casas decimais.

Taxa de acertos (%)			
Trace	**Técnica de predição**		
	NT	**T**	**Direção**
trace_fft.mi	47.12%	52.88%	67.09%
trace_gsm_decoder.me	43.90%	56.10%	65.81%
trace_patricia.mi	42.67%	57.33%	70.12%
Taxa de acertos média	44.56%	55.44%	67.67%

Table 1: Taxas de acerto por técnica de predição

(b) Por que a técnica de predição taken em geral apresenta uma taxa de acertos melhor do que a técnica de predição not-taken?

A técnica de predição taken geralmente apresenta uma taxa de acertos melhor por conta da estrutura dos programas reais. A maioria dos desvios

em um programa real é tomada. Este comportamente é observado devido às várias estruturas de repetições e de tomadas de decisão existentes em um programa. Por exemplo, em um `for(i=0; i<1000; i++)` o desvio é tomado 999 vezes.

4 Experimento 3: Técnicas Dinâmicas

As técnicas de predição dinâmica usam o BPB, um buffer em hardware. Uma estimativa simples do custo do hardware do BPB é:

nº total de bits do BPB = nº linhas BPB × nº de bits em uma linha do BPB

Desejamos analisar as 2 técnicas de predição dinâmica implementadas, predição 1-bit e predição 2-bits, com o trace `trace rijndael encoder mi.txt`.

(a) Para cada técnica de predição dinâmica e cada valor de nLinhasBPB, calcule:

- A taxa de acertos
- O nº total de bits do BPB

As taxas de acerto devem ser apresentadas na forma de porcentagem, arredondadas para 2 casas decimais.

Predição 1-bit								
nLinhasBPB	16	32	64	128	256	512	1024	2048
Taxa de acertos (%)	76.28%	80.27%	84.13%	84.25%	84.31%	84.42%	88.23%	88.23%
nº total de bits do BPB	16	32	64	128	256	512	1024	2048

Predição 2-bits								
nLinhasBPB	16	32	64	128	256	512	1024	2048
Taxa de acertos (%)	76.64%	89.77%	91.96%	92.02%	92.05%	92.11%	94.01%	94.01%
nº total de bits do BPB	32	64	128	256	512	1024	2048	4096

(b) Por que as taxas de acerto melhoram, à medida que o nLinhasBPB cresce?

Ao aumentar o `nLinhasBPB`, o tamanho do Branch Prediction Buffer aumenta. Com um BPB maior, a probabilidade de duas instruções diferentes mapearem para a mesma entrada diminui. Há mais "espaço" para que cada instrução de desvio tenha sua própria entrada dedicada para registrar seu histórico de forma isolada.

(c) Considerando apenas esse trace, se desejamos obter uma taxa de acertos de aproximadamente 90%, com o menor custo possível do hardware, que técnica de predição devemos escolher e qual será o n^o *total de bits do BPB*?

Considerando apenas esse trace, a melhor escolha seria a técnica de predição com 2 bits, com nLinhasBPB igual a 32, já que o número total de bits é 64. Caso a escolha fosse de predição com 1 bit, com nLinhasBPB igual a 1024, a porcentagem de acertos seria menor, e o custo seria 16 vezes maior, pois o número total de bits seria 1024.

(d) Considerando apenas esse trace, se desejamos ter um custo do hardware de no máximo n^o *total de bits do BPB* = 2048, com a maior taxa de acertos possível, que técnica de predição devemos escolher e qual será a taxa de acertos?

Considerando somente esse trace, a melhor escolha seria a técnica de predição com 2 bits (com nLinhasBPB igual a 1024), uma vez que, de acordo com os dados da simulação, obteve o melhor desempenho para o número de bits igual a 2048 com taxa de acertos igual a 94.01%.

5 Experimento 4: Interferência

(a) Descreva detalhadamente quais modificações são necessárias no simulador simpred para que ele calcule a taxa de interferências de cada técnica de predição dinâmica.

Para detectar interferências, uma sugestão de implementação seria algo que identificasse quando duas ou mais instruções de branch diferentes mapeiam para o mesmo índice do BPB. Para isso, poderíamos implementar uma estrutura de dados que associe cada índice do BPB ao endereço da última instrução de branch que o usou. A cada acesso ao BPB, verificar se o endereço atual é diferente do último endereço associado àquela linha do BPB. À frente disso, inicializar um contador de interferências. Por fim, realizar o cálculo da taxa de interferência.

(b) Ao fazer a simulação de uma técnica de predição dinâmica, com um determinador valor de $nLinhasBPB$, como a taxa de interferências pode ser usada para propor alguma mudança?

A taxa de interferência pode ser usada como uma métrica direta dos conflitos do BPB. Se a taxa de interferência for alta, isso sugere que o número de entradas é insuficiente para os padrões de acesso do programa, logo aumentar o número de linhas seria uma solução viável. Outrossim, poderíamos usar essa métrica para implementar técnicas mais sofisticadas de indexação, implementar preditores com tags e, por conseguinte, avaliar essas trocas de técnicas de acordo com suas respectivas taxas de interferência.