《计算机组成与设计》书后作业

姓名:	刘韬
学院:	竺可桢学院
专业:	人工智能
邮箱:	3220103422@zju.edu.cn

日期:	2024-06-10

chapter4

4.1

考虑指令

```
and rd, rs1, rs2
```

4.1.1

RegWrite	ALUSrc	ALUOperation	MemWrite	MemtoReg	MemRead
1	0	and	0	0	0

4.1.2

Registers,ALUsrcmux,ALU,MemToRegmux.

4.1.3

All blocks produce some output. The outputs of DataMemory and Imm Gen are not used.

4.4

4.4.1

Only loads are broken. MemToReg is either 1 or “don’t care” for all other instructions.

4.4.2

I-type, loads, stores.

4.6

4.6.1

no need

4.6.2

Branch	MemRead	MemToReg	ALUop	MemWrite	ALUSrc	RegWrite
0	0	0	add	0	1	1

4.7

4.7.1

R-type:$30 + 250 + 150 + 25 + 200 + 25 + 20 = 700ps$

4.7.2

ld: $30 + 250 + 150 + 25 + 200 + 250 + 25 + 20 = 950ps$

4.7.3

sd:$30 + 250 + 150 + 25 + 200 + 250 = 905ps$

4.7.4

beq:$30 + 250 + 150 + 25 + 200 + 5 + 25 + 20 = 705ps$

4.7.5

I-type:$30 + 250 + 150 + 25 + 200 + 25 + 20 = 700ps$

4.7.6

The longest = 950ps

4.8

$0.52 \times 700ps + 0.25 \times 950ps + 0.11 \times 905ps + 0.12 \times 705ps = 785.65ps$

$950 \div 785.65 = 1.21$

4.13

4.13.1

需要添加几个 MUX

4.13.2

没有需要改造的

4.13.3

There needs to be a path from the ALU output to data memory's write data port. There also needs to be a path from read data 2 directly to Data memory's Address input.

4.13.4

These new data paths will need to be driven by muxes. These muxes will require control wires for the selector.

4.13.5

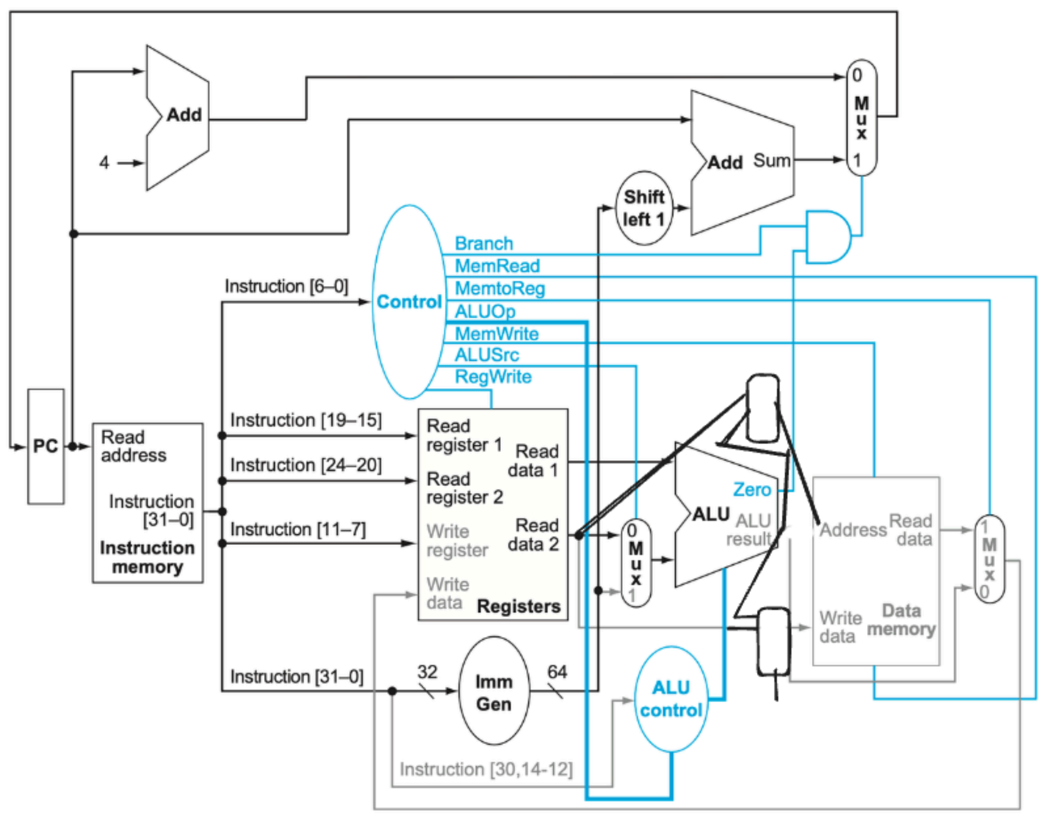


图 1: 4.13.5

4.16

4.16.1

pipeline: 350 ns; single cycle: 1250ns.

4.16.2

pipeline: 1250 ns; single cycle: 1250ns.

4.16.3

Split the ID stage.This reduces the clock-cycle time to 300ps.

4.16.4

Load + Store = 35%

4.16.5

ALU + Load = 65%

4.17

n + k - 1

4.18

x13 = 33, x14 = 26

4.19

x15 = 52

4.20

```
addi x11, x12, 5
NOP
NOP
add x13, x11, x12
addi x14, x11, 15
NOP
add x15, x13, x12
```

4.22.1

```
sd  x29, 12(x16)  IF ID EX ME WB
ld  x29, 8(x16)    IF ID EX ME WB
sub x17, x15, x14    IF ID EX ME WB
bez x17, label         ** ** IF ID EX ME WB
add x15, x11, x14            IF ID EX ME WB
sub x15,x30,x14                IF ID EX ME WB
```

4.23

4.23.1

The clock period won't change because we aren't making any changes to the slowest stage.

4.23.2

Moving the MEM stage in parallel with the EX stage will eliminate the need for a cycle between loads and operations that use the result of the loads. This can potentially reduce the number of stalls in a program.

4.23.3

Removing the offset from ld and sd may increase the total number of instructions because some ld and sd instructions will need to be replaced with a addi/ld or addi/sd pair.

4.24

The second one.We detect the data hazard in the ID stage and stall the pipeline, which can only be done on ID stage.

4.25

4.25.1

… indicates a stall.! indicates a stage that does not do useful work.

```
ld x10, 0(x13)       IF ID EX ME WB
ld x11, 8(x13)          IF ID EX ME WB
add x12, x10, x11          IF ID .. EX ME!WB
addi x13, x13, -16            IF .. ID EX ME!WB
bnez x12, LOOP                   .. IF ID EX ME WB
```

4.25.2

no such cycle.

4.27

4.27.1

```
add x15, x12, x11
nop
nop
ld  x13, 4(x15)
ld  x12, 0(x2)
nop
or  x13, x15, x13
nop
nop
sd  x13, 0(x15)
```

4.27.2

It is not possible to reduce the number of NOPs.

4.27.3

The code executes correctly. We need hazard detection only to insert a stall when the instruction following a load uses the result of the load. That does not happen in this case.

4.27.4

cycle	1	2	3	4	5	6	7	8	
add	IF	ID	EX	ME	WB				
ld		IF	ID	EX	ME	WB			
ld			IF	ID	EX	ME	WB		
or				IF	ID	EX	ME	WB	
sd					IF	ID	EX	ME	WB

1. FowardingA = X; FowardingB = X;
2. FowardingA = X; FowardingB = X;
3. FowardingA = 0; FowardingB = 0;
4. FowardingA = 2; FowardingB = 0;
5. FowardingA = 1; FowardingB = 1;
6. FowardingA = 0; FowardingB = 2;
7. FowardingA = 0; FowardingB = 2;

4.27.5

EX/MEM and MEM/WB rs1,rs2,rd,RegWrite

4.27.6

cycle	1	2	3	4	5	6
add	IF	ID	EX	ME	WB	
ld		IF	ID	-	-	EX
ld			IF	-	-	ID

1. PCWrite = 1;IF/IDWrite = 1;control mux = 0
2. PCWrite = 1;IF/IDWrite = 1;control mux = 0
3. PCWrite = 1;IF/IDWrite = 1;control mux = 0
4. PCWrite = 0;IF/IDWrite = 0;control mux = 1
5. PCWrite = 0;IF/IDWrite = 0;control mux = 1

4.28

4.28.1

$1 + (1 - 0.45) \times 0.25 \times 3 = 1.4125$

4.28.2

$1 + (1 - 0.55) \times 0.25 = 1.1125$

4.28.3

$1 + \times 0.25 = 1.0375$

4.28.4

$1 + (1 - 0.85) \times 0.125 = 1.01875$

$1.0375 \div 1.01875 = 1.0184$

4.28.5

1.125n instructions. CPI = $1 + (1 \div 1.125) \times 0.15 = 1.01665$

$1.0375 \div (1.01665 \times 1.125) = 0.91$

4.28.6

$0.8 \times 1 + 0.2 \times x = 0.85 \Rightarrow x = 0.25$

chapter5

5.2.2

Word Address	binary	Tag	Index	Offset	Hit/Miss
0x03	0000 0011	0	1	1	Miss
0xb4	1011 0100	b	2	0	Miss
0x2b	0010 1011	2	5	1	Miss
0x02	0000 0010	0	1	0	Hit
0xbf	1011 1111	b	7	1	Miss
0x58	0101 1000	5	4	0	Miss
0xbe	1011 1110	b	7	0	Hit
0x0e	0000 1110	0	7	0	Miss
0xb5	1011 0101	b	2	1	Hit
0x2c	0010 1100	2	6	0	Miss
0xba	1011 1010	b	5	0	Miss
0xfd	1111 1101	f	6	1	Miss

5.3.1

block size = 2 word = 16 bytes

32KB data = 32KB / 16B = 2^11 lines

64 bits Address = 3 bits offset + 11 bits index + 1 bit valid + 49 bits tag

The cache is composed of 2^{15} $\times$ 8bits + 2^{11} $\times$ 49tag + 2^{11} $\times$ 1valid = 364544bits = 45568bytes.

5.5

5.5.1

each block has 4 8-bytes words.

5.5.2

index is 5 bits,so there is 32 blocks.

5.5.3

Data is 32×4 words/blocks $\times$ 8bytes/words $= 1024$ bytes $= 8192$ bits

cache is consist of 8192bits of data $+$ (54bits of tags $+$ 1bits of valid) $\times$ 32blocks $=$ 9952 bits

The ratio = 9952 $\div$ 8192 = 1.21

5.5.4

Address	tags	index	offset	Miss/Hit	Replacement
0x00	0x0	0x00	0x00	M	N
0x04	0x0	0x00	0x04	H	N
0x10	0x0	0x00	0x10	H	N
0x84	0x0	0x04	0x04	M	N
0xE8	0x0	0x07	0x08	M	N
0xA0	0x0	0x05	0x00	M	N
0x400	0x1	0x00	0x00	M	0x00-0x1F
0x1E	0x0	0x00	0x1E	M	0x400-0x41F
0x8C	0x0	0x04	0x0C	H	N
0xC1C	0x3	0x00	0x1C	M	0x00-0x1F
0xb4	0x0	0x05	0x14	H	N
0x884	0x03	0x04	0x04	M	0x80-0x9F

5.5.5

$4 \div 12 = 33.3\%$

5.5.6

```
<index, tag, data>
<0, 3, Mem[0xC00]-Mem[0xC1F]>
<4, 2, Mem[0x880]-Mem[0x89f]>
<5, 0, Mem[0x0A0]-Mem[0x0Bf]>
<7, 0, Mem[0x0e0]-Mem[0x0ff]>
```

5.6

5.6.1

The L1 cache has a low write miss penalty while the L2 cache has a high write miss penalty. A write buffer between the L1 and L2 cache would hide the write miss latency of the L2 cache. The L2 cache would benefit from write buffers when replacing a dirty block, since the new block would be read in before the dirty block is physically written to memory.

5.6.2

On an L1 write miss, the word is written directly to L2 without bringing its block into the L1 cache. If this results in an L2 miss, its block must be brought into the L2 cache, possibly replacing a dirty block, which must first be written to memory.

5.6.3

After an L1 write miss,the block will reside in L2 but not in L1.A subsequent read miss on the same block will require that the block in L2 be written back to memory, transferred to L1, and invalidated in L2.

5.10

5.10.1

P1:1.515GHZ; P2:1.11GHZ

5.10.2

$70\text{ns} \div 0.66\text{ns} = 107$ cycles,$1 + 0.08 \times 107 = 9.58$cycles

$70\text{ns} \div 0.90\text{ns} = 78$ cycles,$1 + 0.06 \times 78 = 5.68$cycles

5.10.3

$1 + 0.08 \times 107 + 0.36 \times 0.08 \times 107 = 12.64$

$1 + 0.06 \times 78 + 0.36 \times 0.06 \times 78 = 7.36$

5.10.4

$1 + 0.08 \times (9 + 0.95 \times 107) = 9.85$ worse

5.10.5

$9.85 + 0.36 \times 8.85 = 13.04$

5.10.6

$1 + 0.08[9 + m \times 107] < 9.56 \Rightarrow m < 0.916$

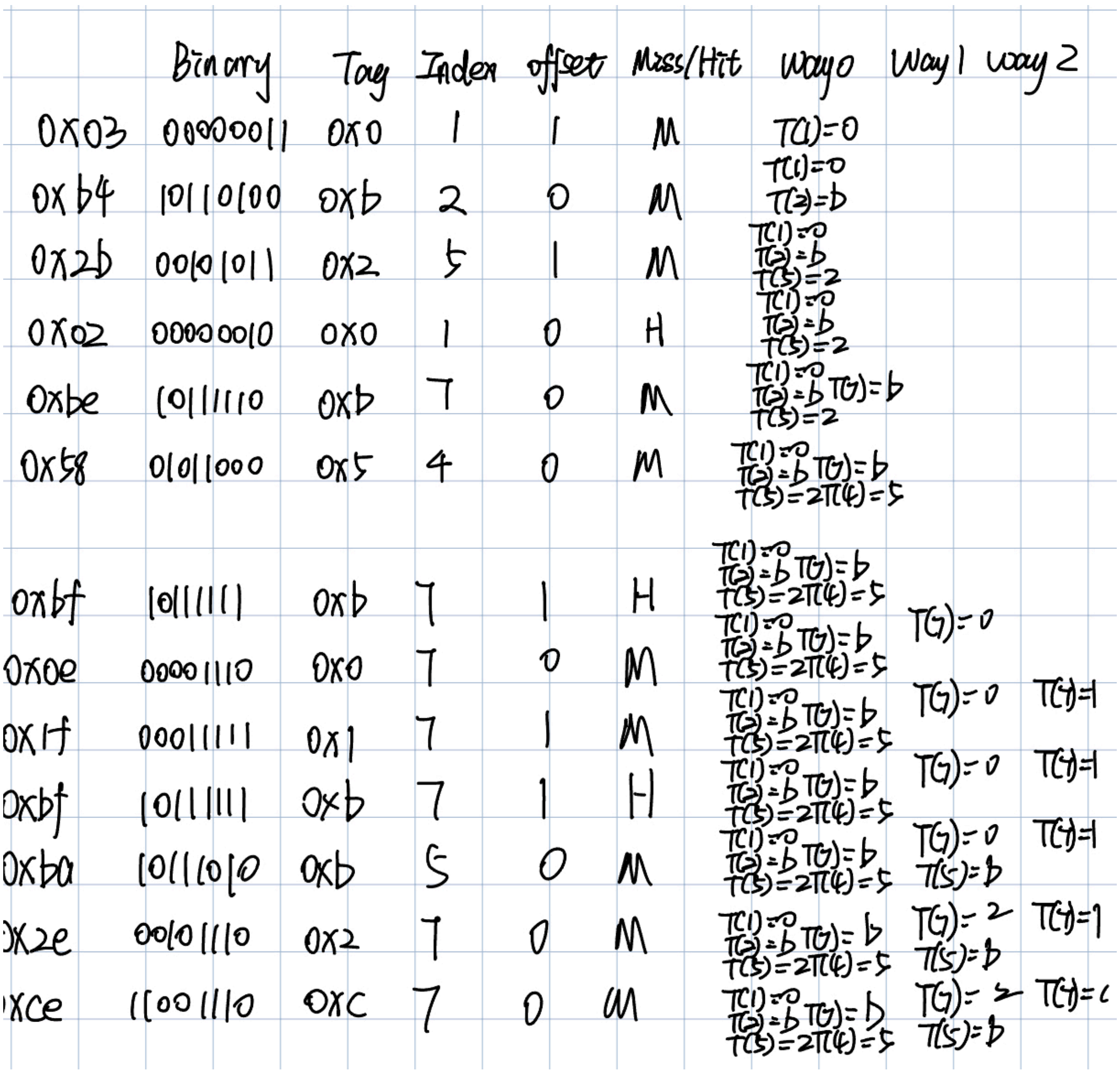
5.10.7

$(\text{CPI\_P1} \times 0.66) < 6.63 \Rightarrow \text{CPI\_P1} < 10.05$

CPI_P1 = AMAT_P1 + 0.36(AMAT_P1 − 1) $\Rightarrow$ AMAT_P1 < 7.65

$1 + 0.08[9 + m \times 107] < 7.65 \Rightarrow m < 0.693$

5.11.2



	Binary	Tag	Index	offset	Miss/Hit	Way0	Way1	way2
0x03	00000011	0x0	1	1	M	T(1)=0		
0xb4	10110100	0xb	2	0	M	T(1)=0 T(2)=b		
0x2b	00101011	0x2	5	1	M	T(1)=0 T(2)=b T(5)=2		
0x02	00000010	0x0	1	0	H	T(1)=0 T(2)=b T(5)=2		
0xbe	10111110	0xb	7	0	M	T(1)=0 T(2)=b T(7)=b T(5)=2		
0x58	01011000	0x5	4	0	M	T(1)=0 T(2)=b T(7)=b T(5)=2 T(4)=5		
						T(1)=0 T(2)=b T(7)=b T(5)=2 T(4)=5		
0xbf	10111111	0xb	7	1	H	T(1)=0 T(2)=b T(7)=b T(5)=2 T(4)=5		
0x0e	00001110	0x0	7	0	M	T(1)=0 T(2)=b T(7)=b T(5)=2 T(4)=5	T(7)=0	
0x1f	00011111	0x1	7	1	M	T(1)=0 T(2)=b T(7)=b T(5)=2 T(4)=5	T(7)=0	T(7)=1
0xbf	10111111	0xb	7	1	H	T(1)=0 T(2)=b T(7)=b T(5)=2 T(4)=5	T(7)=0	T(7)=1
0xba	10111010	0xb	5	0	M	T(1)=0 T(2)=b T(7)=b T(5)=2 T(4)=5	T(7)=0 T(5)=b	T(7)=1
X2e	00101110	0x2	7	0	M	T(1)=0 T(2)=b T(7)=b T(5)=2 T(4)=5	T(7)=2 T(5)=b	T(7)=1
Xce	11001110	0xc	7	0	M	T(1)=0 T(2)=b T(7)=b T(5)=2 T(4)=5	T(7)=2 T(5)=b	T(7)=c

图 2:

5.13

5.13.1

3 years + 1 day

5.13.2

$1095 \times 1096 = 99.91\%$

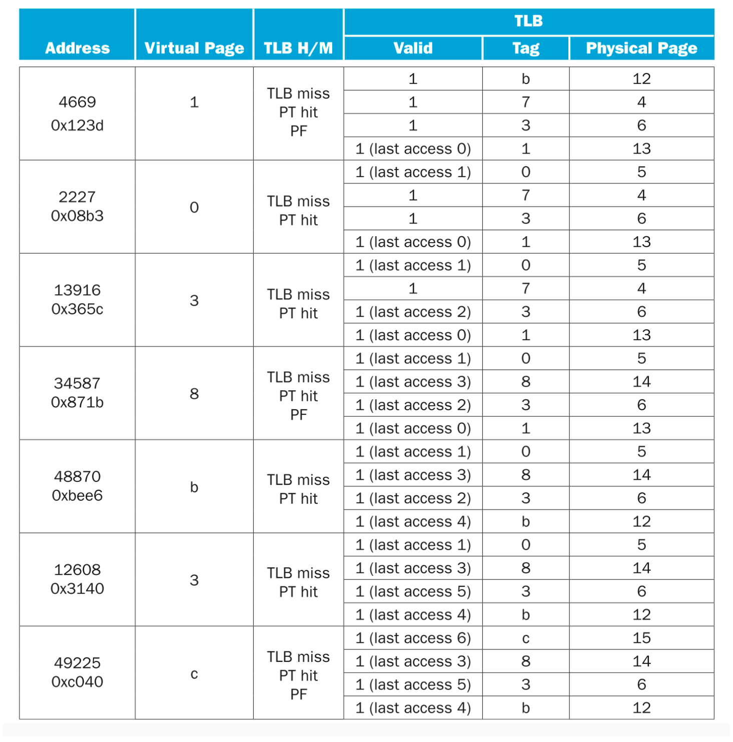
5.13.3

Availability approaches 1.0.

5.13.4

MTTR becomes the dominant factor in determining availability. However, availability would be quite high if MTTF also grew measurably. If MTTF is 1000 times MTTR, it the specific value of MTTR is not significant.

5.16.1



Address	Virtual Page	TLB H/M	TLB		
			Valid	Tag	Physical Page
4669 0x123d	1	TLB miss PT hit PF	1	b	12
			1	7	4
			1	3	6
			1 (last access 0)	1	13
2227 0x08b3	0	TLB miss PT hit	1 (last access 1)	0	5
			1	7	4
			1	3	6
			1 (last access 0)	1	13
13916 0x365c	3	TLB miss PT hit	1 (last access 1)	0	5
			1	7	4
			1 (last access 2)	3	6
			1 (last access 0)	1	13
34587 0x871b	8	TLB miss PT hit PF	1 (last access 1)	0	5
			1 (last access 3)	8	14
			1 (last access 2)	3	6
			1 (last access 0)	1	13
48870 0xbee6	b	TLB miss PT hit	1 (last access 1)	0	5
			1 (last access 3)	8	14
			1 (last access 2)	3	6
			1 (last access 4)	b	12
12608 0x3140	3	TLB miss PT hit	1 (last access 1)	0	5
			1 (last access 3)	8	14
			1 (last access 5)	3	6
			1 (last access 4)	b	12
49225 0xc040	c	TLB miss PT hit PF	1 (last access 6)	c	15
			1 (last access 3)	8	14
			1 (last access 5)	3	6
			1 (last access 4)	b	12

图 3:

5.17

5.17.1

tag size = 32 − log 2(8192) = 19 All five page tables would require 5 ×(2^19 ×4) bytes = 10 MB.

5.17.2

second level has $2^{19-8} = 2048$ entries. requiring 2048 × 4 = 8 KB each and covering 2048 × 8 KB = 16 MB (2^24) of the virtual address space.

If we assume that "half the memory" means 2^31 bytes, then the minimum amount of memory required for the second-level tables would be 5 × (2^31/2^24)×8 KB = 5 MB. The first-level tables would require an additional 5 × 128 × 6 bytes = 3840 bytes.

The maximum amount would be if all 1st-level segments were activated, requiring the use of all 256 segments in each application. This would require 5 × 256 × 8 KB = 10 MB for the second-level tables and 7680 bytes for the first-level tables.

5.17.3

The page index is 13bits(address bits 12 down to 0). A 16 KB direct-mapped cache with two 64-bit words per block would have 16-byte blocks and thus 16 KB/16 bytes = 1024 blocks. Thus, it would have 10 index bits and 4 offset bits and the index would extend outside of the page index. The designer could increase the cache's associativity. This would reduce the number of index bits so that the cache's index fits completely inside the page index.

5.20

5.20.1

There are no hits.

5.20.2

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0
M	M	M	M	M	M	M	M	H	H	M	M	M	M	H	H	M

5.20.3

5.20.4

MRU is an optimal policy.

5.20.5

The best block to evict is the one that will cause the fewest misses in the future. Unfortunately, a cache controller cannot know the future! Our best alternative is to make a good prediction.

5.20.6

If you knew that an address had limited temporal locality and would conflict with another block in the cache, choosing not to cache it could improve the miss rate. On the other hand, you could worsen the miss rate by choosing poorly which addresses to cache.