22b Cache Examples – Part 2
CSC 230

Department of Computer Science
University of Victoria

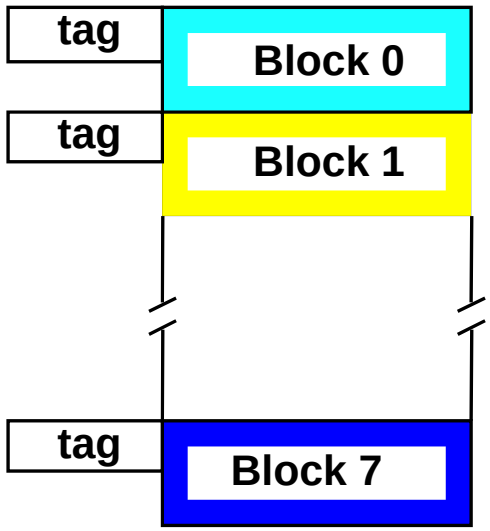
Problem 5.6 (H textbook)

A program consists of two nested loops – a small inner loop and a much larger outer loop (see below). The decimal memory addresses delineate the locations of the two loops and the program. All locations in the sections (e.g. 17-22, 23-264) contain sequential instructions. Cache and memory organized as in previous example.

Main memory = 64K words

Cache size = 1K words

Block size = 128 words



Cache

each block = 128 words

total 1K words

Think it through again (see ex. 5.6)

How are the total bits organized in a direct-mapped cache with:

Main memory = 64K words

Cache size = 1K words

Block size = 128 words

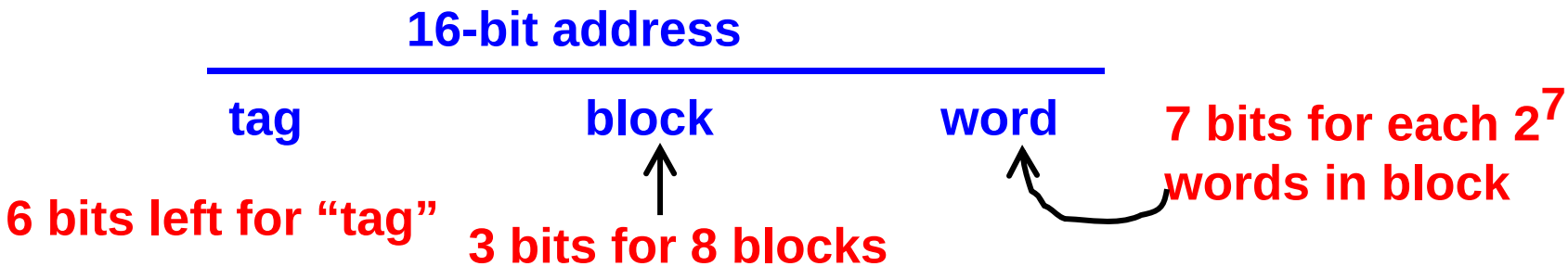
Memory = 64 K words = 2^6 x 2^{10} words = 2^{16} words

Cache = 1 K words = 2^{10} words

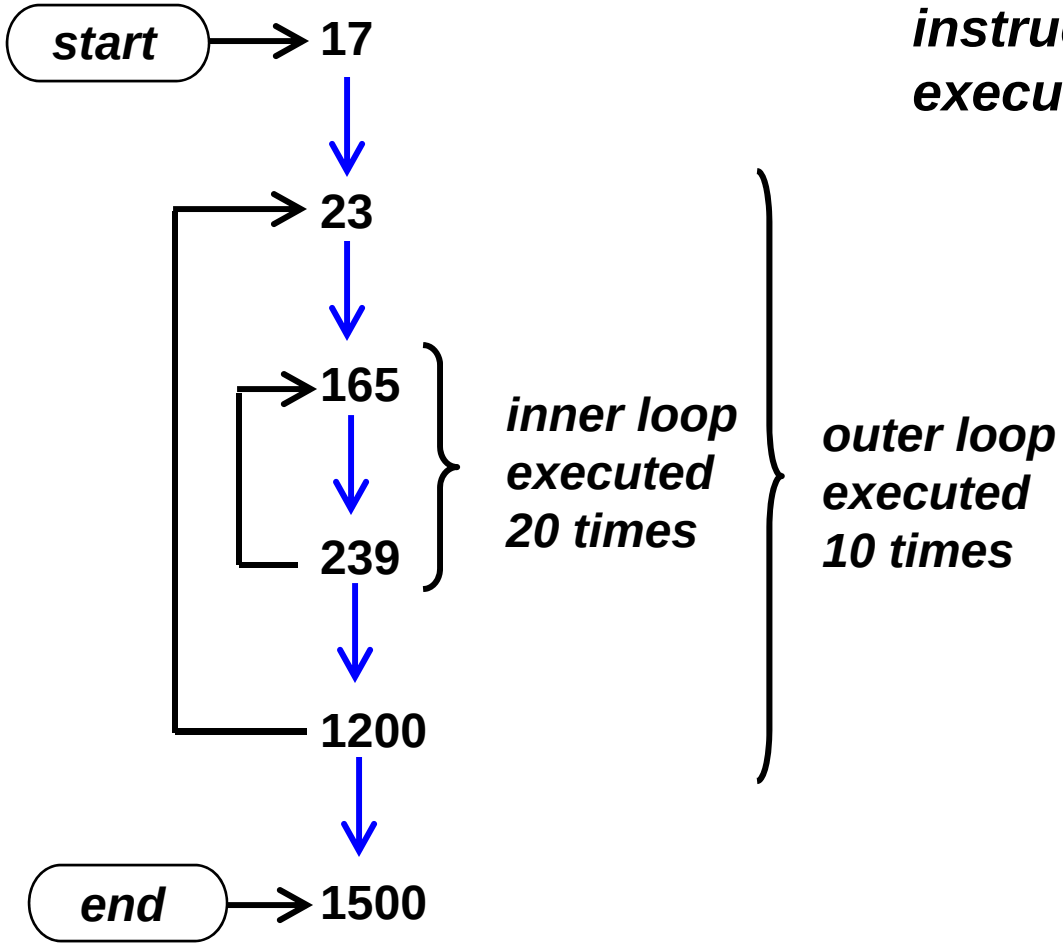
since block size = 128 words = 2^7 words, then:

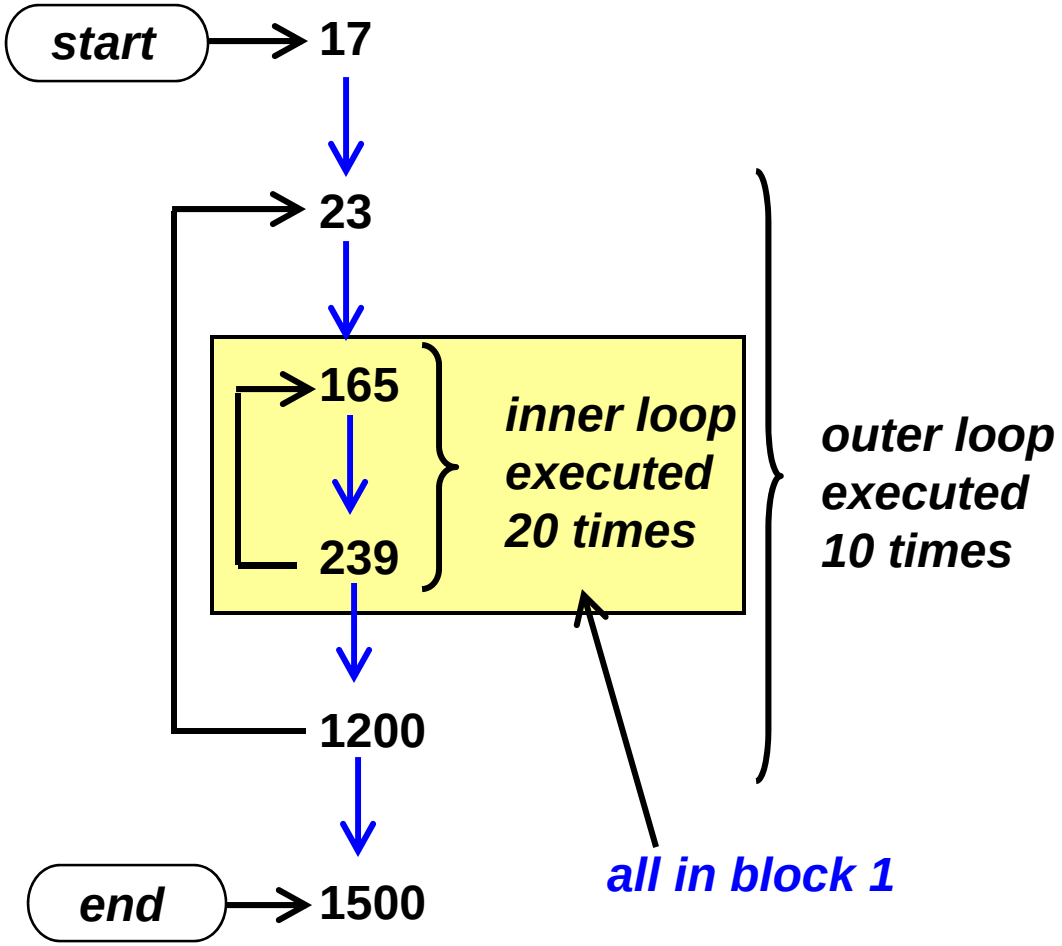
2^{10} words / 2^7 words = 2^3 ➔ 8 blocks in cache

thus cache has 8 blocks, each containing 128 words

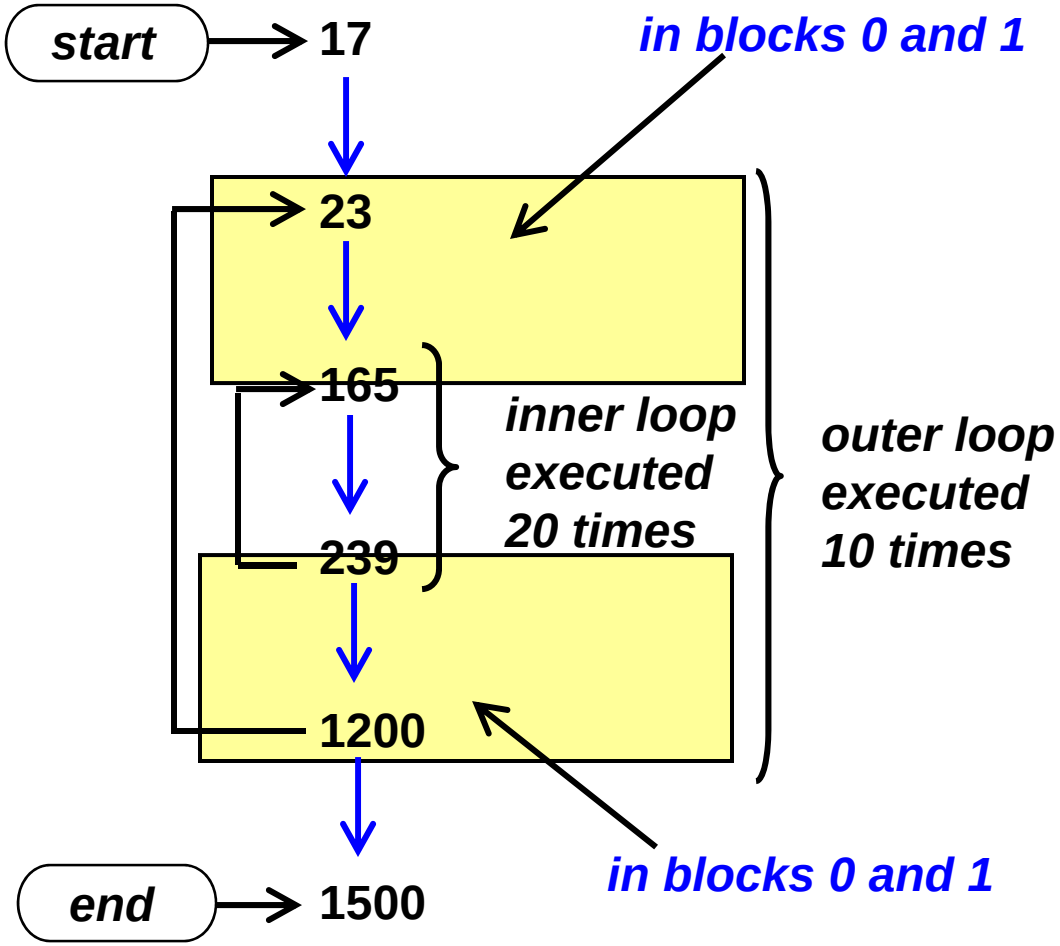


Compute total time needed for instruction fetching during execution





Block 0	0	1024
	127	1151
Block 1	128	1152
	255	1279
Block 2	256	1280
	383	1407
Block 3	384	1408
	511	1535
Block 4	512	
	639	
Block 5	640	
	767	
Block 6	768	
	895	
Block 7	896	
	1023	

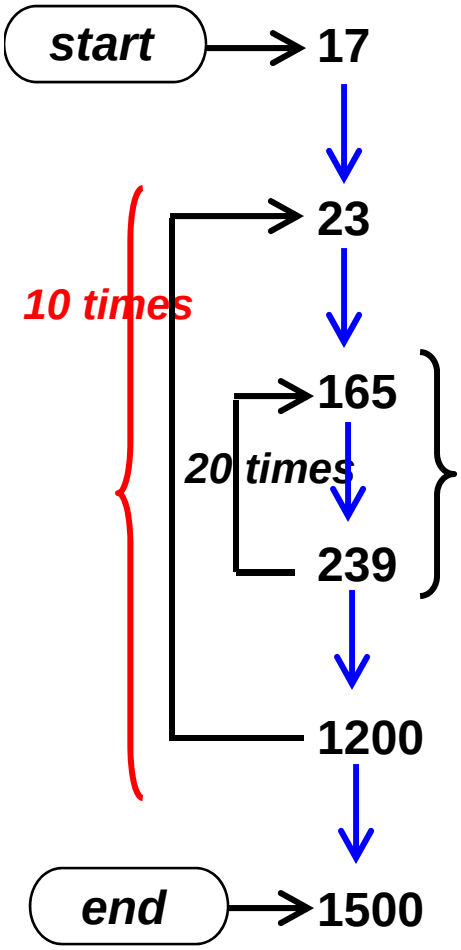


Block 0	0	1024
	127	1151
Block 1	128	1152
	255	1279
Block 2	256	1280
	383	1407
Block 3	384	1408
	511	1535
Block 4	512	
	639	
Block 5	640	
	767	
Block 6	768	
	895	
Block 7	896	
	1023	

Blocks 0 and 1 get overwritten, 2-7 remain

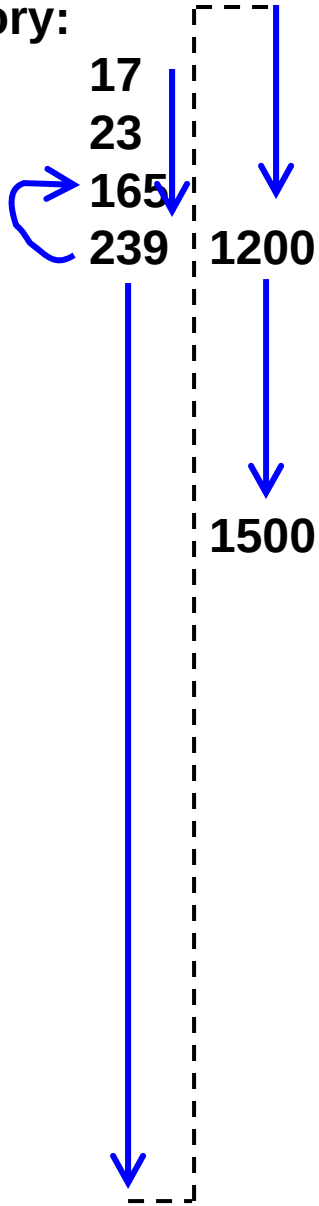
Sequence of blocks read from memory:

[1] 0,1,2,3,4,5,6,7,0,1



then:

[2] 0,1,0,1
[3] 0,1,0,1
[4] 0,1,0,1
.....
[8] 0,1,0,1
[9] 0,1,0,1
[10] 0,1,0,1,2,3



Block 0	0	1024
	127	1151
Block 1	128	1152
	255	1279
Block 2	256	1280
	383	1407
Block 3	384	1408
	511	1535
Block 4	512	
	639	
Block 5	640	
	767	
Block 6	768	
	895	
Block 7	896	
	1023	

Sequence of blocks read from memory:

memory access 10 t, cache 1 t

[1] 0,1,2,3,4,5,6,7,0,1

then:

[2] 0,1,0,1

[3] 0,1,0,1

[4] 0,1,0,1

[5] 0,1,0,1

[6] 0,1,0,1

[7] 0,1,0,1

[8] 0,1,0,1

[9] 0,1,0,1

[10] 0,1,0,1,2,3

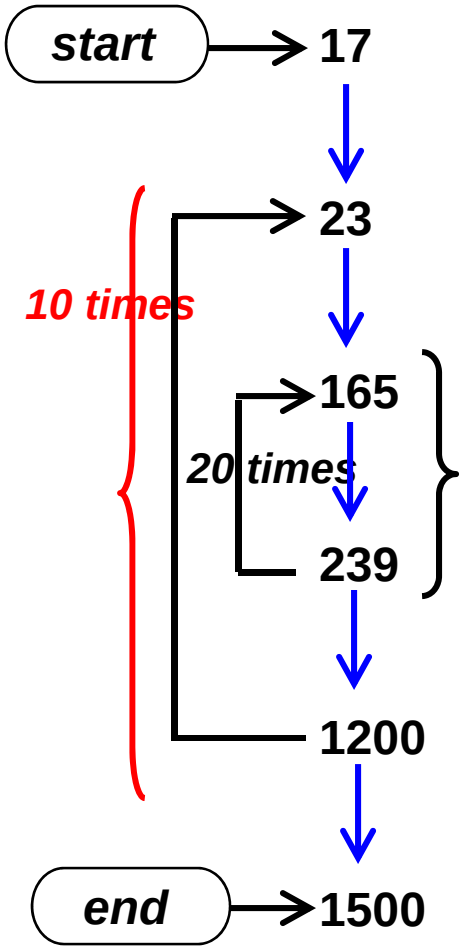
[1] = 10

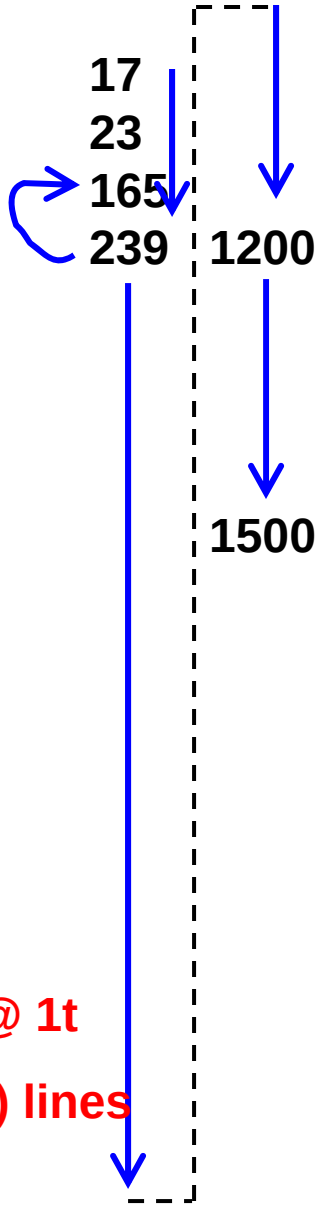
[2] – [9] = 4 x 8

[10] = 6

(48 t) x 128 words x 10 t = 61,440 t

➔ time to read from memory into cache





Block 0	0	1024
	127	1151
Block 1	128	1152
	255	1279
Block 2	256	1280
	383	1407
Block 3	384	1408
	511	1535
Block 4	512	
	639	
Block 5	640	
	767	
Block 6	768	
	895	
Block 7	896	
	1023	

inner loop = (239-164) x 200 times @ 1t

outer – inner = (1200-22) – (239-164) lines

x 10 times @ 1t

end part = (1500-1200) x 1 time @ 1t

(48 t) x 128 words x 10 t = 61,440 t

➔ time to read from memory into cache

inner loop = (239-164) x 200 times @ 1t = 11,030 t

outer – inner = (1200-22) – (239-164) lines

x 10 times @ 1t = 15,000 t

end part = (1500-1200) x 1 time @ 1t = 300 t

TOTAL = 61,440 + (11,030 + 15,000 + 300) = 87,770

NOTE: if no cache = 61,440 + 263,300 = 324,740

with cache only 27% of time