CS 354 - Machine Organization & Programming
Thursday, October 17, 2019

Project p3 (6%): DUE at 10 pm on Monday, October 28th

Homework hw3 (1.5%): DUE TOMORROW at 10 pm on Friday, October 18th

Last Time

- Footers
- Explicit Free List
- Explicit Free List Improvements
- Heap Caveats
- Memory Hierarchy

Today

- Memory Hierarchy (from last time)
- Locality (from last time)
- Bad Locality
- Rethinking Addressing
- Caching Basic Idea

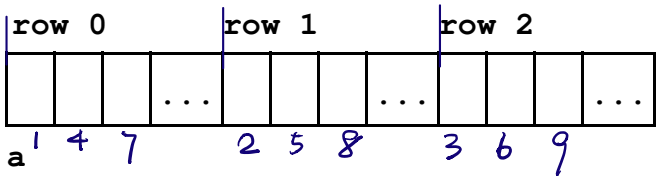
Next Time

- Designing Caches & Varying Set Size
- **Read:** B&O 6.4.3 - 6.4.4

Bad Locality

Why is this code bad?

```
int a[ROWS][COLS];
      ≈3
for (int c = 0; c < COLS; c++)
   for (int r = 0; r < ROWS; r++)
      a[r][c] = r * c;
```



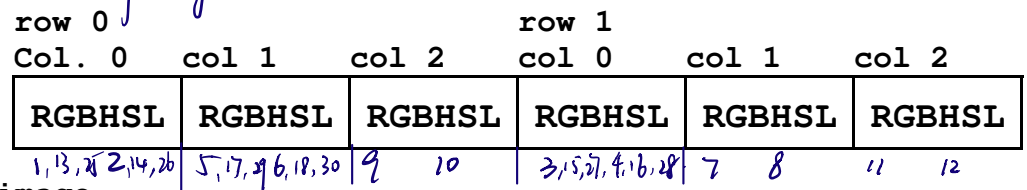
c = a. 1, 2 jumping around the mem.

→ How would you improve the code to reduce stride? Flip c & r loops.

Key Questions for Determining Spatial Locality:

1. what does mem layout looks like.
2. what is the stride of your code.

Why is this code bad?




```
struct {
   float rgb[3];
   float hsl[3];
} image[HEIGHT][WIDTH];
      ≈2      ≈3

for (int v = 0; v < 3; v++)
   for (int c = 0; c < WIDTH; c++)
      for (int r = 0; r < HEIGHT; r++) {
         image[r][c].rgb[v] = 0;
         image[r][c].hsl[v] = 0;
      }
```

v = 0, 1.

➢ How would you improve the code to reduce stride?

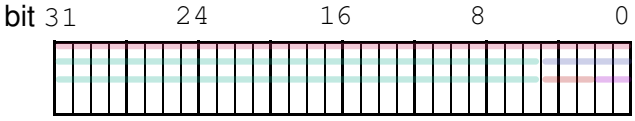
Good or bad locality?

- Instruction Flow:
  - sequencing? Good spatial.
  - selection? Bad. (jumping around)
  - repetition? Good temporal. & spacial.
- Searching Algorithms:
  - linear search: array, good spatial
    linked list. bad
  - binary search: array bad

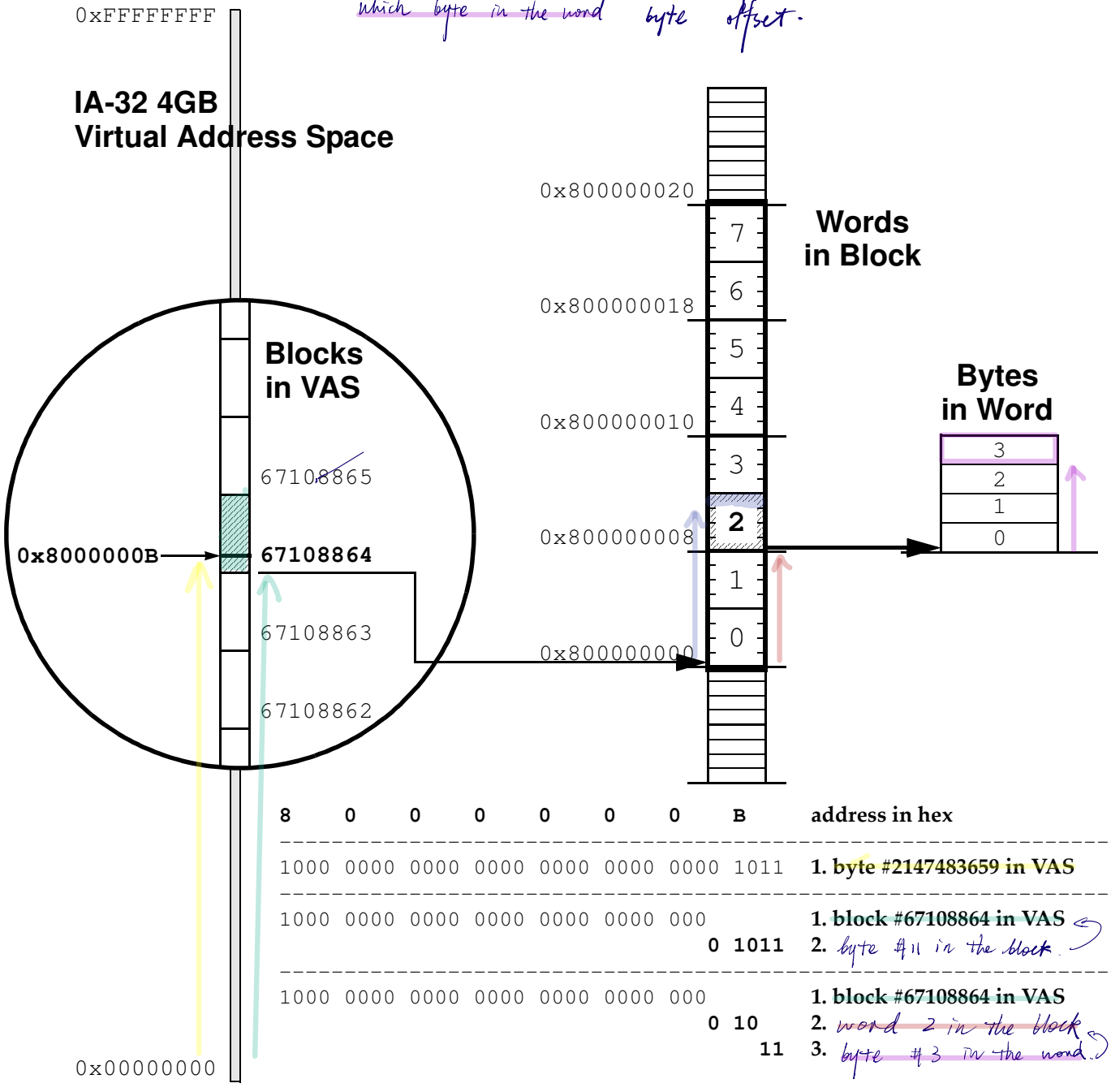
❋ *Programs with good locality*

Rethinking Adressing

- *An address identifies* which type in virtual address space (VAS) to access.
- *An address can be* divided into parts. to access mem in multiple steps.



cache block is 32 bytes in IA-32
step 1. which block in VAS
step 2. which byte in block
word is 4 bytes in IA-32 which word in the block word offset
which byte in the word byte offset.



8	0	0	0	0	0	0	B	address in hex
1000	0000	0000	0000	0000	0000	0000	1011	1. byte #2147483659 in VAS
1000	0000	0000	0000	0000	0000	000		1. block #67108864 in VAS
							0 1011	2. byte #11 in the block
1000	0000	0000	0000	0000	0000	000		1. block #67108864 in VAS
							0 10	2. word 2 in the block
							11	3. byte #3 in the word

Basic Caching Idea

Assume memory is divided into 32 byte blocks, and all needed blocks are already in main memory.
Cache L1 has 4 locations to store blocks and L2 has 16 locations to store blocks.

Consider the CPU accessing the following blocks in this sequence:

22,11,22,44,11,33,11,22,55,27,44

4 hit 4 hit

cache miss
when a block isn't found in this cache so fetch from next lower level.

cold miss when a cache location is empty or invalid

capacity miss when cache is too small for the working set.

conflict miss when 2 or more blocks map to share location

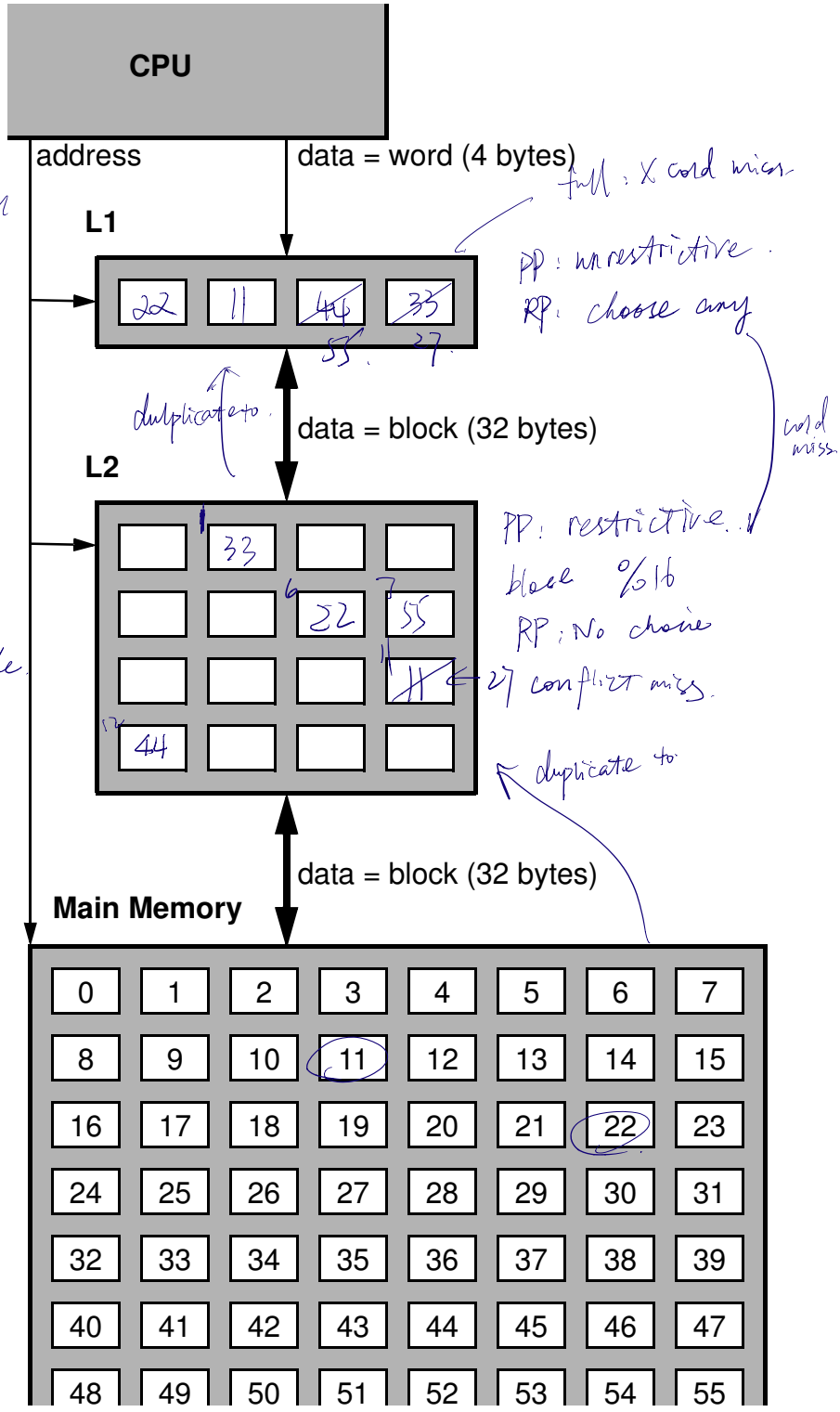
cache hit faster memory access. when a block is found in the cache.

placement policies
1. unrestrictive.
2. restrictive.

replacement policies
1. choose any location.
2. no choice

victim block cache block chosen to be replaced.

working set group of blocks used by a process during some time interval.



full: X cold miss.
FP: unrestrictive.
RP: choose any
duplicate to
cold miss
FP: restrictive.
block %16
RP: No choice
27 conflict miss.
duplicate to