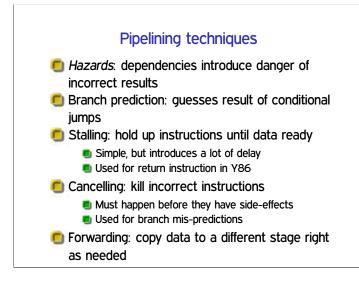


## **Pipelining basics**

- Split processing into stages, and work on multiple instructions at once
- Reduces cycle time and increases hardware utilization
- Pipeline registers hold data between stages
- Performance concerns: balanced stages, and not too many
- Correctness concerns: must have same final behavior



## Outline

Topics in CPU architecture

Topics in code optimization

Topics in memory hierarchy and caches

**Discussion problems** 



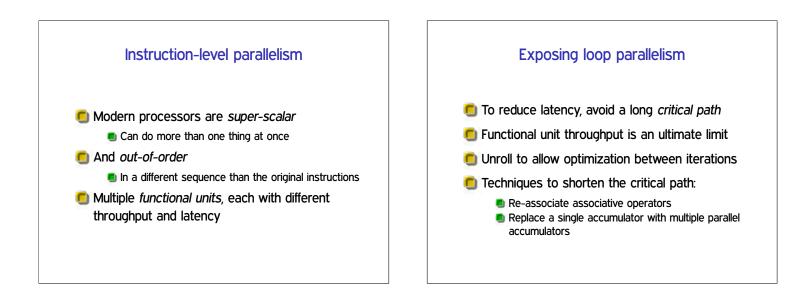
Concentrate on the program parts that run the most

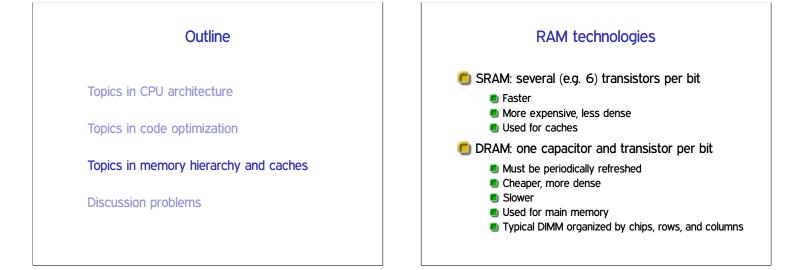
- Amdahl's law bounds possible speedup
- Array-style programs: concentrate on inner loops
- Complex programs: use a profiler
- Know what the compiler can and can't do
  - Compiler can be smart, but is careful about correctness
  - Functions and pointers (aliasing) block optimization
- Watch out for algorithmic problems

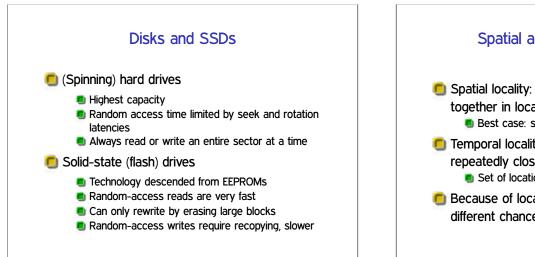
## Machine-independent optimizations



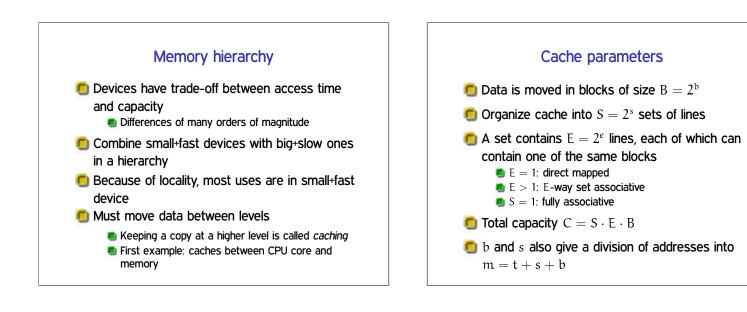
- Avoid abstract functions in time-critical code
- Use temporary variables to reduce memory operations
- Unroll loops to reduce bookkeeping overhead

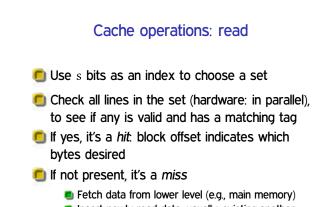


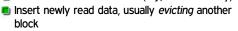


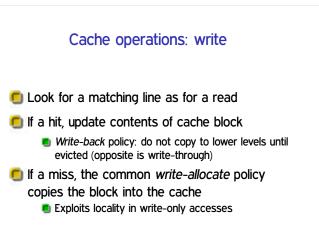












## Cache usage optimizations

- Overall goals: maximize locality, minimize working set
- Use more compact data representations
- 🆲 Prefer stride-1 data accesses
  - E.g., for a matrix, iterate over indexes in outer-to-inner order
- Temporally group accesses to the same data values
  - For 2-D data, group by blocks (tiles) instead of rows

## Outline

Topics in CPU architecture

Topics in code optimization

Topics in memory hierarchy and caches

### **Discussion problems**

## Y86 "compiling"

int ary[10][10]; ary[i][j]++;

ary is in %eax, i is in %ebx, j is in %ecx. Step 1: write a formula for &ary[i][j]

## Y86 "compiling"

int ary[10][10]; ary[i][j]++;

ary is in %eax, i is in %ebx, j is in %ecx. Step 1: write a formula for &ary[i][j]

4\*(j + 10 \* i) + ary

# **Y86 "compiling", pt. 2** ary **is in** %eax, i **is in** %ebx, j **is in** %ecx. 4\*(j + 10 \* i) + ary

```
rrmovl %ebx, %esi # esi = i
addl %esi, %esi # esi = 2*i
      %esi, %esi # esi = 4*i
addl
addl
      %ebx, %esi # esi = 5*i
addl
      %esi, %esi # esi = 10*i
addl
      %ecx, %esi # esi = 10*i + j
addl
      %esi, %esi # esi = 2*(10*i + j)
addl
      %esi, %esi # esi = 4*(10*i + j)
addl
      %eax, %esi # esi = ary + 4*(10*i + j)
```

# Y86 "compiling", pt. 3 Instructions for (\*%esi)++

## Y86 "compiling", pt. 3

Instructions for (\*%esi)++

mrmovl	O(%esi), %edi	# Load into %edi
irmovl	1, %edx	
addl	%edx, %edi	# %edi++
rmmovl	%edi, O(%esi)	<pre># Store back</pre>

### Optimization Why does the following program run slowly? char \*concat(char \*a, char \*b) { char \*c = malloc(strlen(a) + strlen(b) + 1); strcpy(c, a); strcat(c, b); free(a); free(b); return c; } int main(int argc, char \*\*argv) { char \*buf = strdup(""); char \*linebuf = 0; size\_t len = 0; int i; while (getline(&linebuf, &len, stdin) != -1) buf = concat(buf, strdup(linebuf)); for (i = strlen(buf) - 1; i >= 0; i--) putchar(buf[i]); return 0; }

## Cache parameters

The following caches all have 64-byte blocks:

	C	Ε	S
Α.	32 KB	1	512
В.	32 KB	8	64
С.	32 KB	512	1

Which cache needs the most gates?

Which cache has the fastest hit time?

Which cache has the lowest miss rate?

Which cache is found in a real Core i7?

# Cache parameters

The following caches all have 64-byte blocks:

	С	Е	S
А.	32 KB	1	512
В.	32 KB	8	64
C.	32 KB	512	1

Which cache needs the most gates? C

Which cache has the fastest hit time?

Which cache has the lowest miss rate?

Which cache is found in a real Core i7?

Cache parameters	Cache parameters		
he following caches all have 64-byte blocks:	The following caches all have 64-byte blocks:		
C E S	C E S		
A. 32 KB 1 512	A. 32 KB 1 512		
B. 32 KB 8 64	B. 32 KB 8 64		
C. 32 KB 512 1	C. 32 KB 512 1		
<ul> <li>Which cache needs the most gates? C</li> <li>Which cache has the fastest hit time? A</li> </ul>	<ul> <li>Which cache needs the most gates? C</li> <li>Which cache has the fastest hit time? A</li> </ul>		
Which cache has the lowest miss rate?	Which cache has the lowest miss rate? C		
Which cache is found in a real Core i7?	Which cache is found in a real Core i7?		

## Cache parameters

The following caches all have 64-byte blocks:

	С	E	S
Α.	32 KB	1	512
B.	32 KB	8	64
C.	32 KB	512	1

Which cache needs the most gates? C
 Which cache has the fastest hit time? A

- Which cache has the lowest miss rate? C
- Which cache is found in a real Core i7? B