RISC versus CISC

Homework 1 released!

You may have noticed…

RISC-V: Reduced Instruction Set Computer

Is there such a thing as a *complex* instruction set computer?

CISC: generally describes x86 and other Intel-produced instruction sets

Outline

- Brief history of Intel (according to Patterson and Hennessey with my opining)
- Motivating the movement back to RISC
- RISC v CISC trade-off
- Not so RISC-y RISC and not so CISC-y CISC

History of Intel*

Intel 8086 architecture announced with an assembly language. 16-bit dedicated registers

1978

Intel 8087 floating point coprocessor. First stack-based Intel architecture

1980

Intel 80386 introduced with 32 bit registers. Modern Intel processors compatible with this instruction set (sometimes called i386)

1985

Intel adds ~130 new instructions to their processors, which must remain backwards compatible with earlier processors. 70 of these were added between 1997 and 1999

1989-1999

“Intel added yet another 144 instructions”

2001

Intel licenses the IA-32 instruction set to AMD, who produces processors with 64 bit register widths. The ISA expands to account for “long mode” instructions

2003

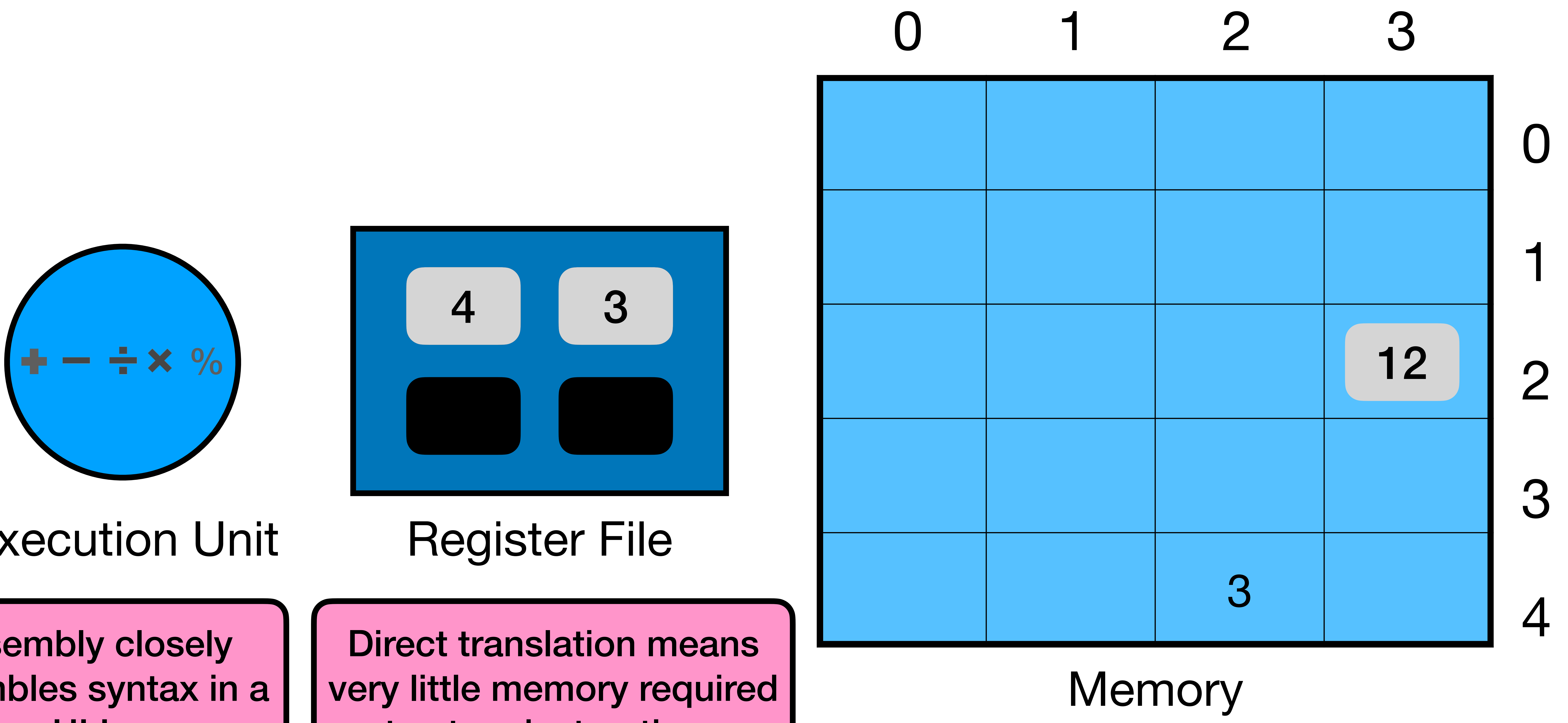
What are the takeaways?

- x86-64 is a *complex instruction set computer* (CISC) that is backwards compatible with early processor models from the 70s and 80s
- The instruction set has evolved to account for more sophisticated software
- x86 remains one of the (if not the) most prevalent architectures in commodity systems!

CISC by Example

```
x *= y;
```

MULT 2:3, 4:2



	0	1	2	3
0				
1				
2				12
3				
4			3	

Instructions operate directly on memory system

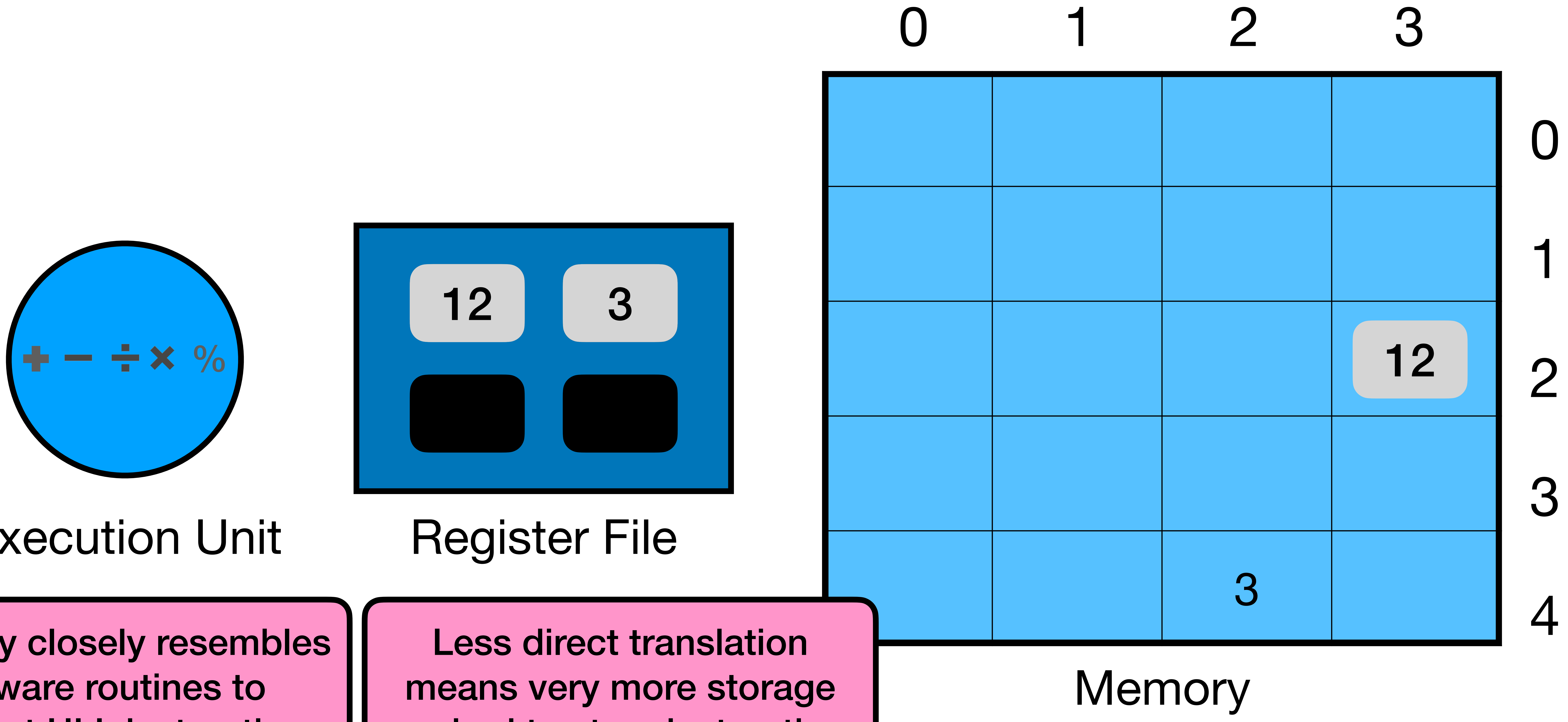
Assembly closely resembles syntax in a HLL

Direct translation means very little memory required to store instructions

RISC by Example

```
x *= y;
```

```
LOAD 2:3, A
LOAD 4:2, B
PROD A, B
STORE 2:3, A
```



Instructions operate directly on memory system

Assembly closely resembles hardware routines to implement HLL instructions

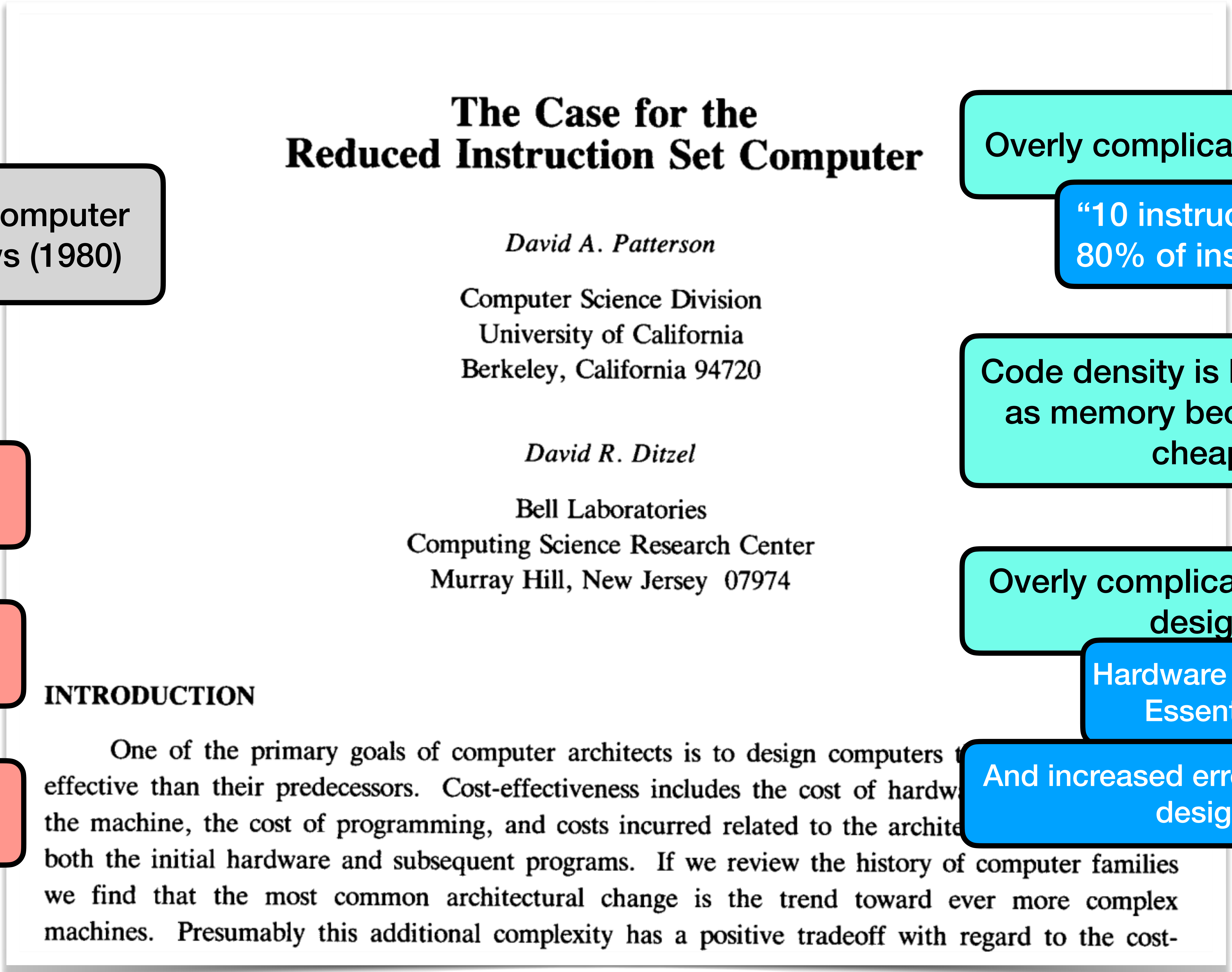
Less direct translation means very more storage required to store instructions

ACM SIGARCH Computer Architecture News (1980)

Marketing!

Support for HLLs

Multiprogramming



The Case for the Reduced Instruction Set Computer

David A. Patterson

Computer Science Division
University of California
Berkeley, California 94720

David R. Ditzel

Bell Laboratories
Computing Science Research Center
Murray Hill, New Jersey 07974

INTRODUCTION

One of the primary goals of computer architects is to design computers t
effective than their predecessors. Cost-effectiveness includes the cost of hardw
the machine, the cost of programming, and costs incurred related to the archite
both the initial hardware and subsequent programs. If we review the history of computer families we find that the most common architectural change is the trend toward ever more complex machines. Presumably this additional complexity has a positive tradeoff with regard to the cost-

Overly complicated compilers

“10 instructions accounted for 80% of instructions executed”

paper

Code density is less important as memory becomes faster cheaper

Overly complicated hardware designs!

Hardware design cycle is ~3 years. Essentially a fixed constant

And increased errors in hardware designs!

The Case for RISC*

We will unpack this later this week!

- Ultimately, a RISC based instruction set requires less complexity to decode instructions (fewer of them) and implement their execution (simpler hardware primitives)

- Simplicity in the design means that a comparable design requires less on-chip space

- Complexity of a program is pushed to *software* (in many cases, the compiler) to support a high level programming language

Make an argument for a CISC instruction set. Do the same for RISC. Which do you find more compelling?

Do you buy Patterson and Ditzel’s (cynical) explanation of the CISC boom during the 70s and 80s? Does your perspective change knowing that the authors also founded the RISC-V project?

One of the arguments in favor of RISC is the reduced complexity, but a HLL compiler to a RISC instruction set is pretty complex. Are you bothered by this contradiction?

Ethical aside…

x86 is everywhere!

It was designed around the constraints of the technology and application of the time, but almost all of these features are obsolete!

Remember: our design decisions have consequences! Don’t be afraid to revisit the validity of original principles!

Not so {R, C}ISC-y

- In practice, many CISC assembly instructions are assembled into “microcodes” in the byte code
- RISC-V has an array of ISA extensions to more efficiently perform certain common functionalities that aren’t explicitly described in the standard ISA
- CISC is becoming more RISC-y!
- RISC is becoming more CISC-y!

Takeaways