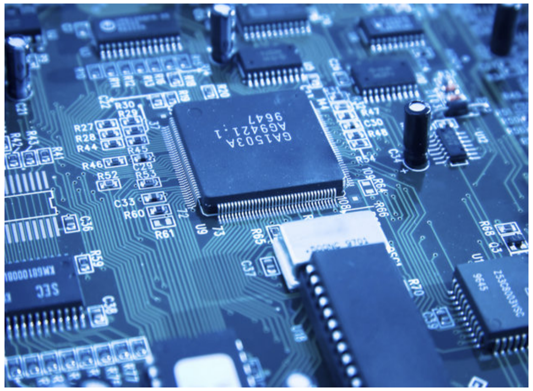
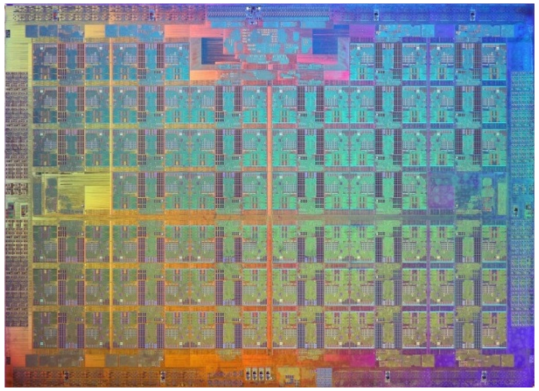
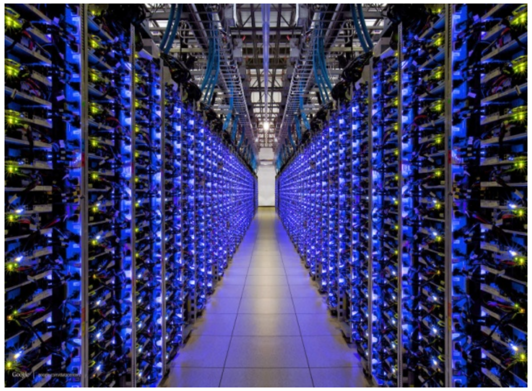
Welcome to 6.004!

Computation Structures

Spring 2019

6.004 Course Staff

Instructors

Arvind
arvind@csail.mit.edu

Silvina Hanono Wachman
silvina@mit.edu

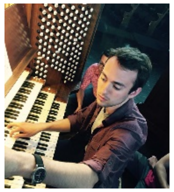
Jason Miller
jasonm@csail.mit.edu

Teaching Assistants

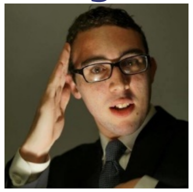
Brian Wheatman

Domenic Nutile

Intae Moon


Kade Phillips

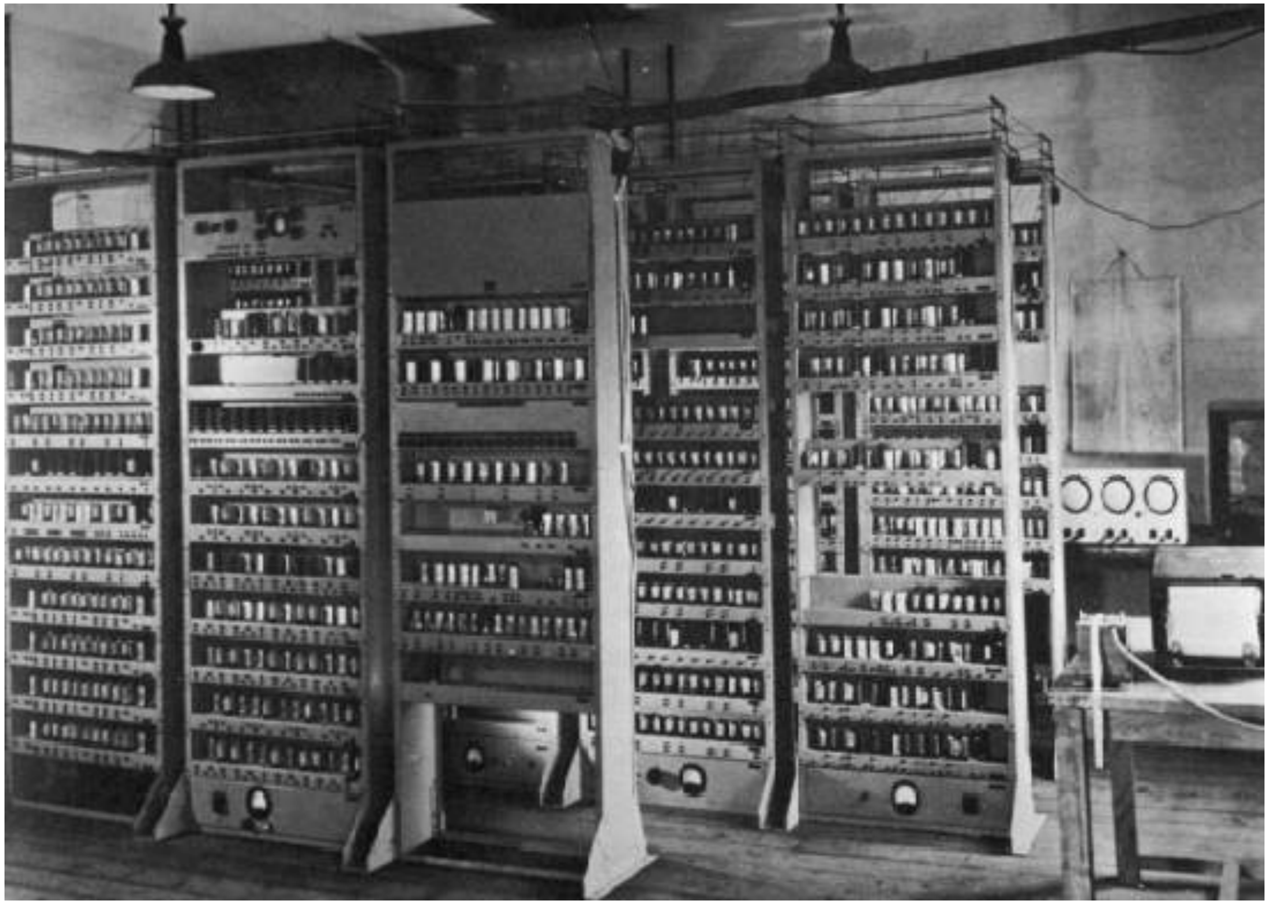
Jeremy Wright



Udgam Goyal

Driss Hafdi

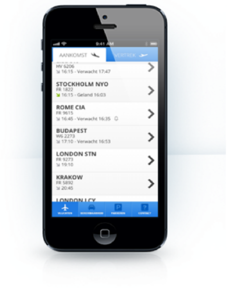
Computing Devices Then…



ENIAC, 1943   30 tons, 200KW, ~1000 ops/sec

Computing Devices Now

Typical 2018 laptop
1kg, 10W, 10 billion ops/s



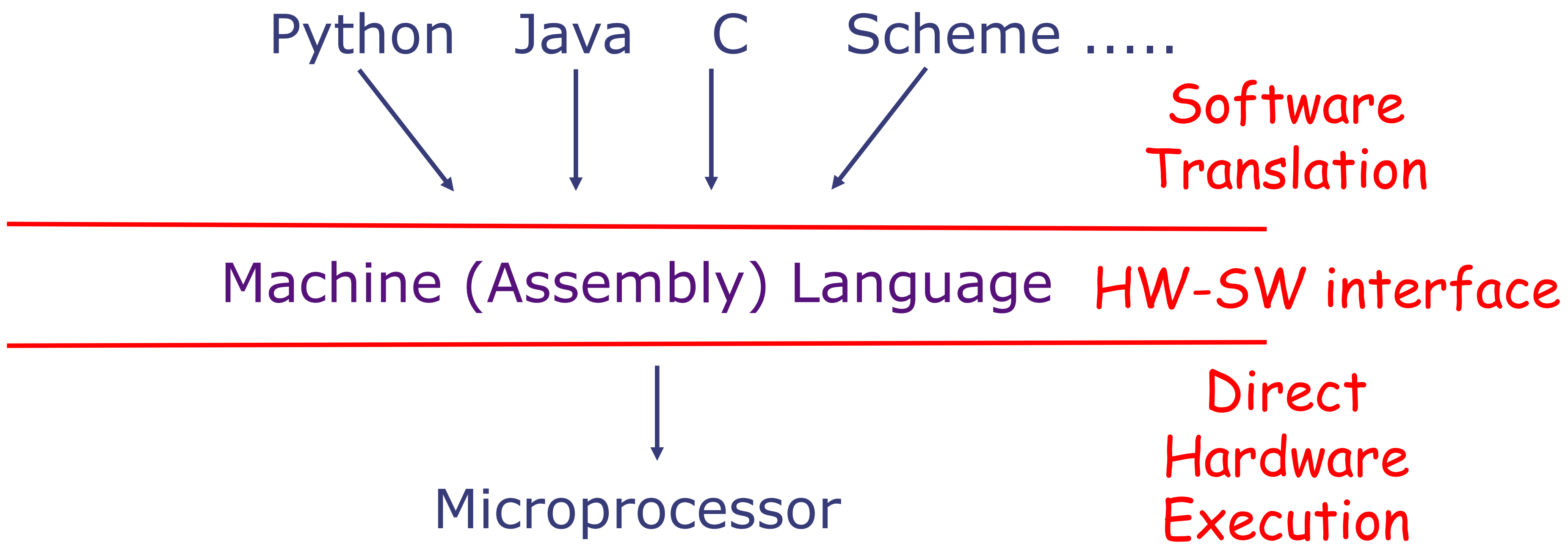
Our Focus: Design of General-Purpose Processors

- Microprocessors are the basic building block of computer systems
  - Understanding them is crucial even if you do not plan to work as a hardware designer
- Microprocessors are the most sophisticated digital systems that exist today
  - Understanding them will help you design all kinds of hardware

By the end of the term you would have designed *many* RISC-V processors from scratch!

“General Purpose” Processor

- It would be highly desirable if the same hardware could execute programs written in Python, Java, C, or for that matter in any high-level language
- It is also not sensible to execute every feature of a high-level language directly in hardware

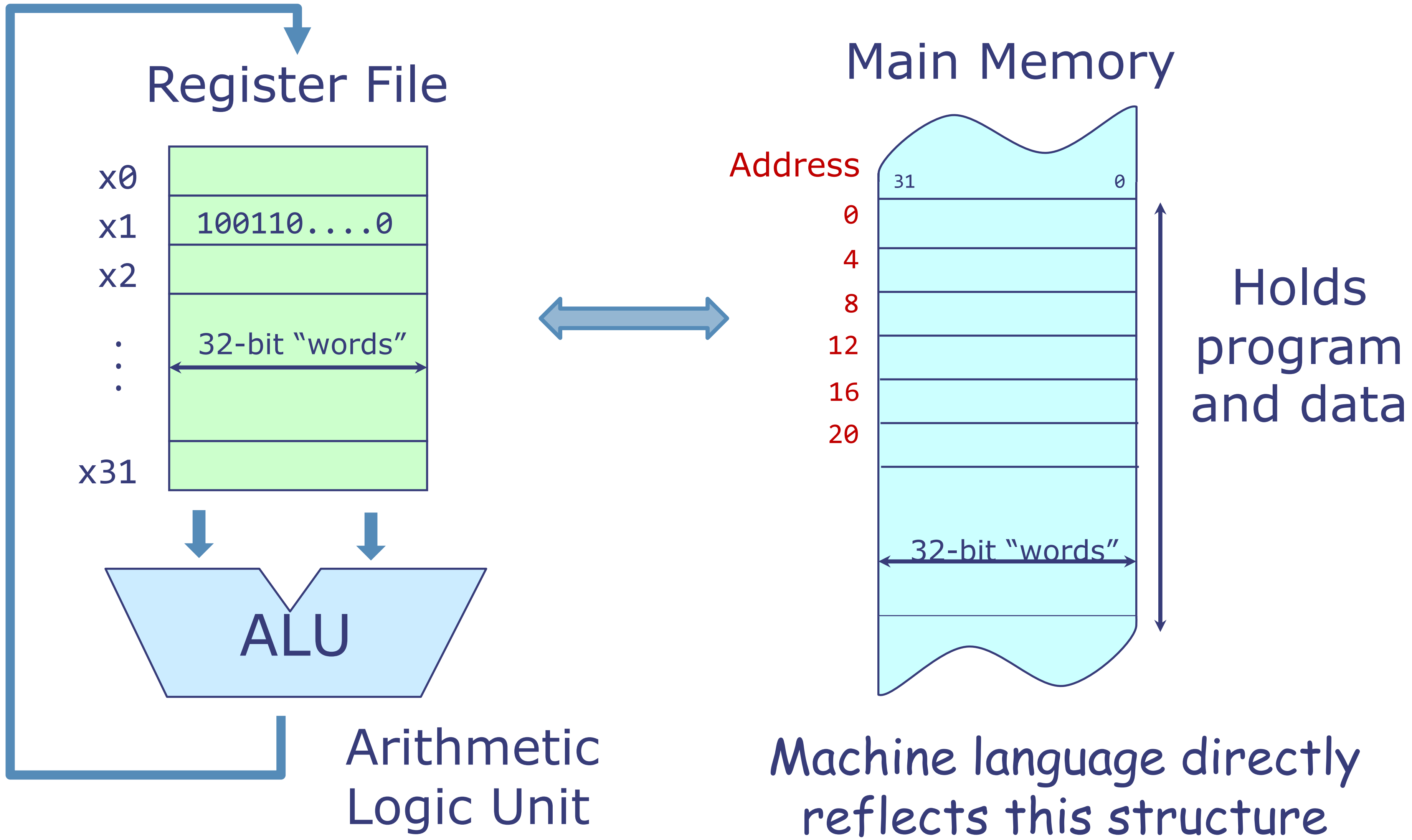


Some facts

- All modern electronic computers are digital, i.e., compute with 1s and 0s
- All the information is converted into binary form and all operations are carried out on these binary forms
- It is possible to implement simple operations like +, >, AND, etc. on binary numbers in hardware super efficiently

Technology dictates hardware: *if tomorrow silicon-based computers were replaced by DNA-based computers, we will have to redesign 6.004 from scratch*

Components of a MicroProcessor



Machine language directly reflects this structure

MicroProcessor Structure / Assembly Language

- Each register is of fixed size, say 32 bits
- The number of registers are small, say 32
- ALU directly performs operations on the register file, typically
  - $x_i \leftarrow \mathrm{Op}(x_j , x_k)$ where $\mathrm{Op} \in \{+, \mathrm{AND}, \mathrm{OR}, <, >, \ldots\}$
- Memory is large, say Giga bytes, and holds program and data
- Data can be moved back and forth between Memory and Register File
  - Ld x M[a]
  - St M[a] x

Assembly (Machine) Language Program

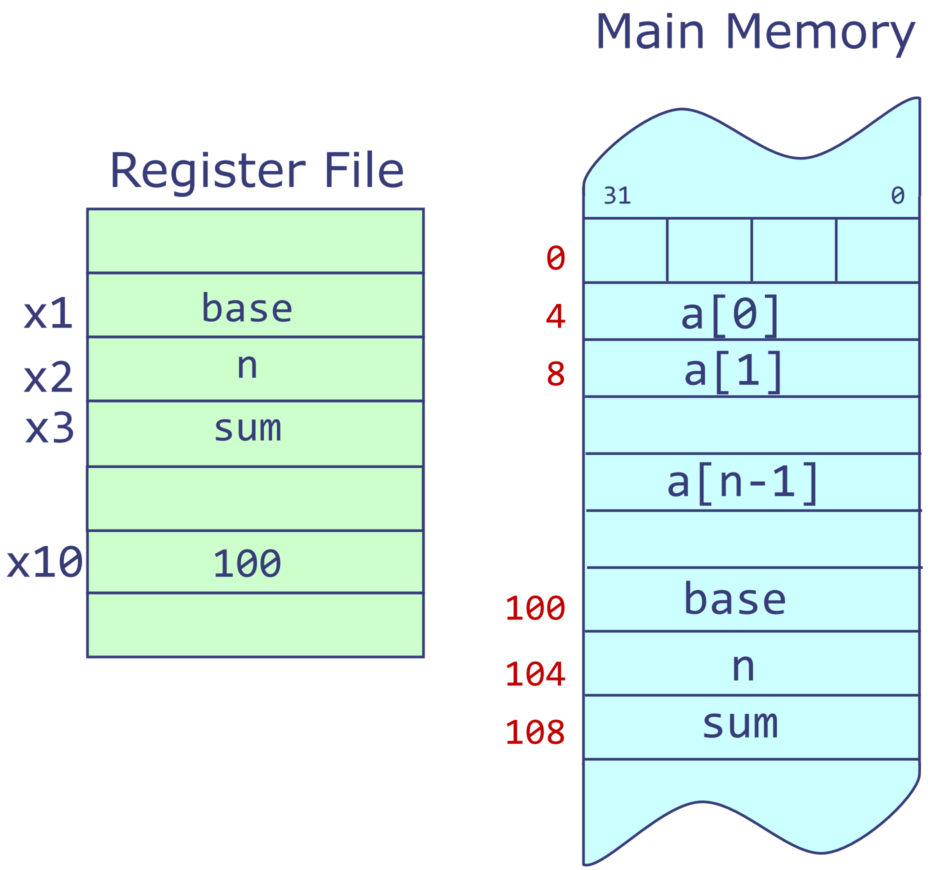
- An assembly language program is a sequence of instructions which execute in a sequential order unless a control transfer instruction is executed
- Each instruction specifies one of the following operations:
  - ALU or Reg-to-Reg operation
  - Ld
  - St
  - Control transfer operation: e.g., xi < xj go to label l

Program to sum array elements

`sum = a[0] + a[1] + a[2] + ... + a[n-1]`

```
   x1 ← load(Mem[x10])
   x2 ← load(Mem[x10+4])
   x3 ← load(Mem[x10+8])
loop:
   x4 ← load(Mem[x1])
   add x3, x3, x4
   addi x1, x1, 4
   addi x2, x2, -1
   bnez x2, loop

   store(Mem[x10+8]) ← x3
```



High Level vs Assembly Language

High Level Language

1. Primitive Arithmetic and logical operations
2. Complex data types and data structures
3. Complex control structures - Conditional statements, loops and procedures
4. Not suitable for direct implementation in hardware

Assembly Language

1. Primitive Arithmetic and logical operations
2. Primitive data structures – bits and integers
3. Control transfer instructions
4. Designed to be directly implementable in hardware

tedious programming!

The course outline

- Module 1
  - Binary representation and operations
  - RISC-V ISA   A new and cool ISA!
  - Assembly language programming in RISC-V
  - Expressing high-level programming in assembly language
  - Procedure calling convention
- Module 2
  - Digital abstraction
  - Boolean algebra and combinational logic
  - Sequential Logic
  - Expressing logic designs in Bluespec, a modern hardware design language
  - Logic synthesis
  - Pipelined and folded circuits

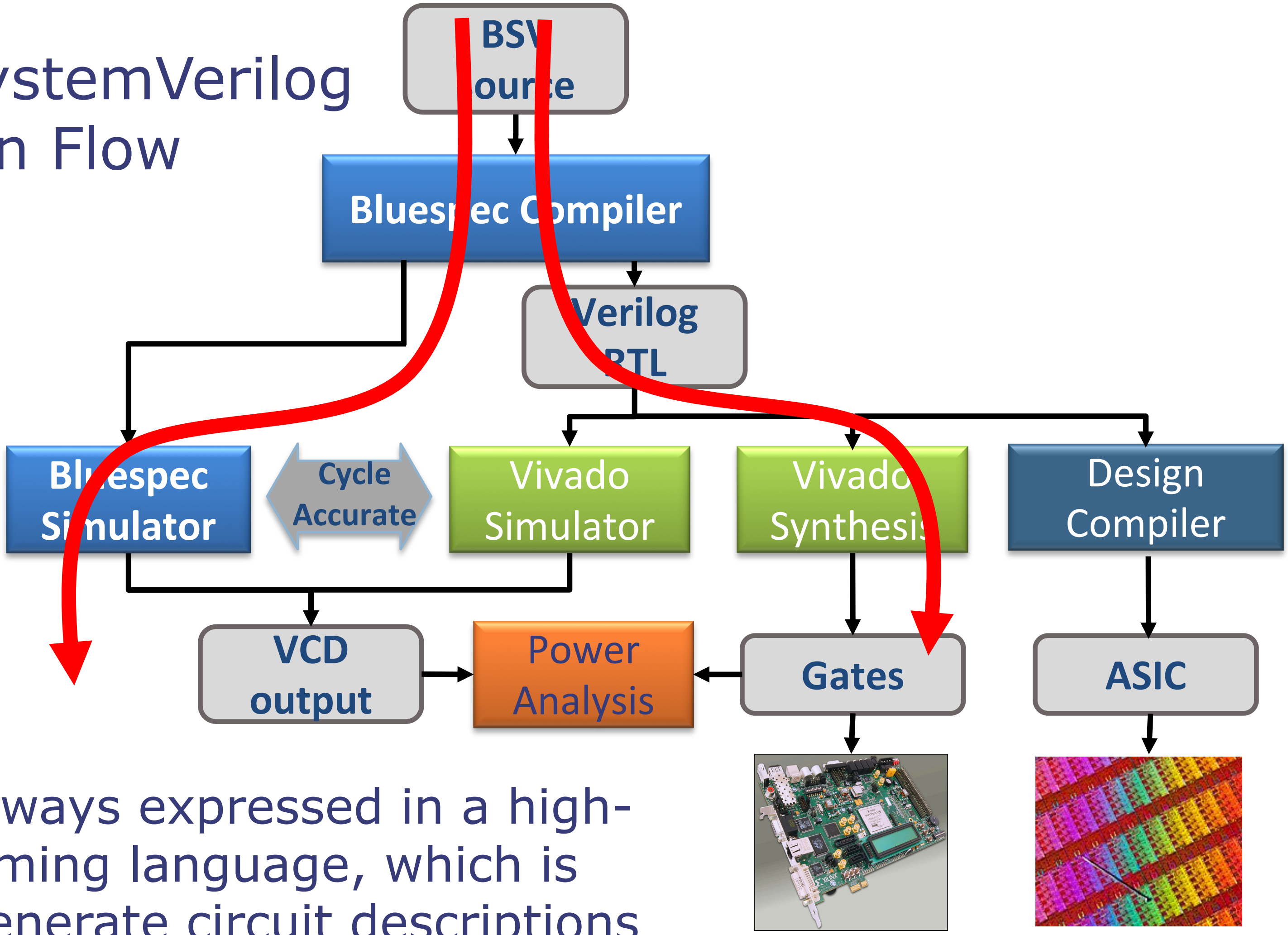
Bluespec is based on modern language design principles

The course outline - continued

- Module 3
  - Implement non-pipelined RISC-V computer
  - Caches
  - Implement Pipelined RISC-V computers
    - control and data hazards, bypasses
  - Branch prediction (time permitting)
- Module 4
  - Operating systems
  - I/O interrupts and exceptions
  - Virtual memory
- Module 5
  - parallel programming and multicore issues – synchronization, cache coherence, ...

We Rely on Modern Design Tools

Bluespec SystemVerilog
Design Flow



Designs are always expressed in a high-level programming language, which is compiled to generate circuit descriptions

6.004 Experience: RISC-V using Bluespec and Yosys Synthesis (Shuotao Xu)

	Yosys synthesis		Quicksort	Quicksort
Processor Design	Clock (ps)	Area(µm^2)	Cycles	CPI
2Stage	599	23475	106161	1.5983
2StagePlus	617	23770	95946	1.4445
2StageDS	669	24234	95946	1.4445
2StageEhr	817	23485	88161	1.3273
3Stage	474	28209	111403	1.6772
3StageEhr	592	27650	102160	1.5380
3StageBypasses	492	27926	108592	1.6349
3StageBypassesBTB	479	35064	99810	1.5027
3StageBypassesEhr	601	27542	97183	1.4631
3StageBypassesEhrBTB	756	36815	92794	1.3970
4Stage	494	26266	111904	1.6847
4StageBypasses	490	26644	91293	1.3744
4Stage2Bypasses	491	27368	86854	1.3076
4rStage2BypassesBTB	493	34411	78892	1.1877
4Stage2BypassesEhrBTB	754	34721	72408	1.0901

Course Mechanics

- Use the course website: http://6004.mit.edu
  - It has up-to-date information, handouts, and references to supplemental reading
- Use Piazza extensively: piazza.com/mit/spring2019/6004
  - Please sign up immediately
  - All major course announcements are made on piazza
  - Fastest way to get your questions answered
  - Please fill out the poll about lecture handouts

Course Logistics

- 2 lectures/week: handouts, videos, and reference materials on website
- 2 recitations/week: work through tutorial problems using skills and concepts from previous lecture

- 8 mandatory lab exercises
  - Online submission + check-off meetings in lab
  - Due throughout the term (7 free late days meant to cover short illnesses etc., see website)
- One open-ended design project
  - Due at the end of the term
- 3 quizzes: March 7, April 11, May 9 (7:30-9:30pm)
  - If you have a conflict, contact us to schedule a makeup

Recitation Mechanics

- 10 recitation sections on Wed & Fri
  - R11 and R12 Cancelled (2 and 3 PM in 8-205).
  - If you have a conflict with your assigned section, or were assigned to R11 or R12, choose a different one—no need to let us know
- Recitation attendance is mandatory (worth 5% of your grade)
  - Everyone has 4 excused absences
- Each lecture has a short take-home problem that you can to try to confirm your understanding of the lecture
- We will provide tutorial problems for lectures when appropriate; these will be reviewed in the recitations

Grading

- 80 points from labs,
  20 points from design project,
  90 points from quizzes,
  10 points from recitation attendance
- Last term’s cutoffs:
  - 165 <= A
  - 145 <= B < 165
  - 125 <= C < 145
  - F: Do not complete all the labs
- This term’s cutoffs will be assigned around the time of the design project

Online and Offline Resources

- The course website: http://6004.mit.edu
  - also contains pointers to useful notes and reference material
- Piazza forum: piazza.com/mit/spring2019/6004
- We will hold regular office hours in the lab (room 32-083) to help you with lab assignments, infrastructure, and any other questions

*32-083 Combination Lock:*__________________

We Want Your Feedback!

- Your input is crucial to fine-tune this offering of the course and improve future versions
- Periodic informal surveys
- Any time: Email us or post on Piazza

Take home

- What is the difference between a calculator and a computer?
- How would you express your name in binary notation?

Thank you!

Next lecture:
Binary Representation and assembly programming