Lecture 1: Course Introduction

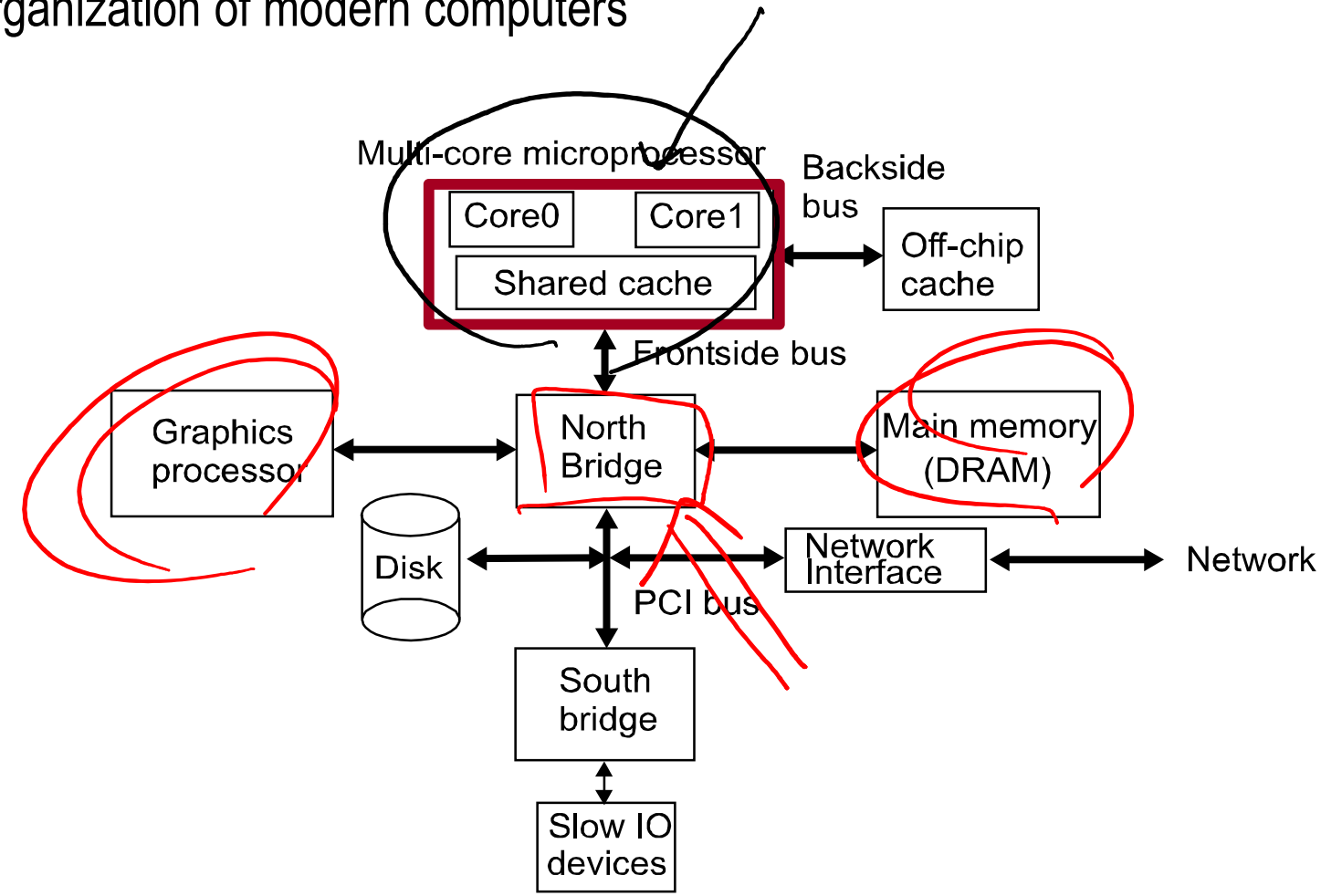
Course overview

Computer is everywhere!

http://images.slideplayer.com/26/8674558/slides/slide_3.jpg

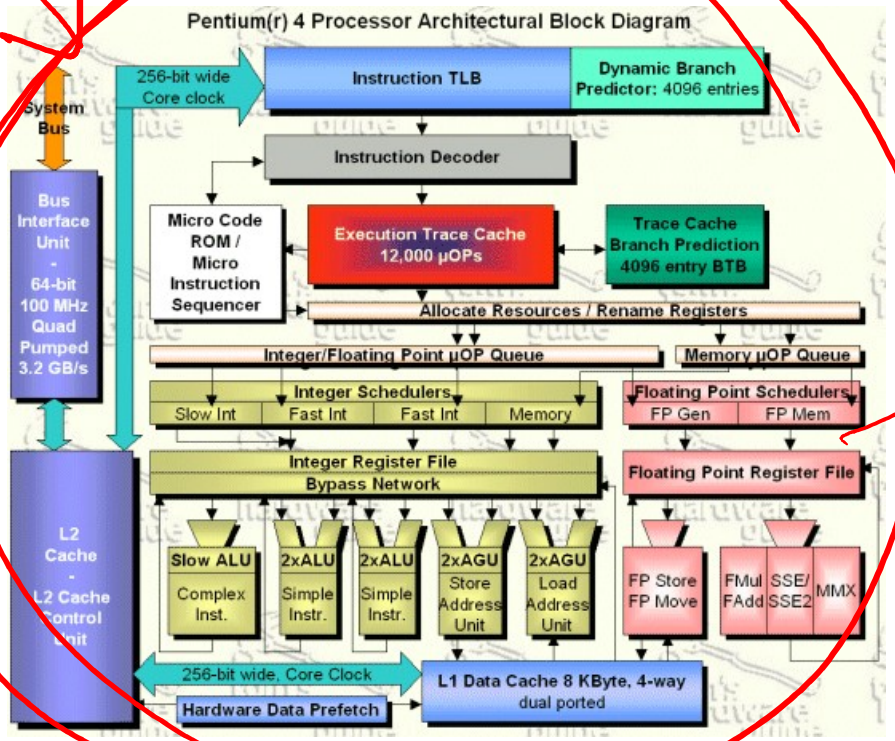
Computer Organization

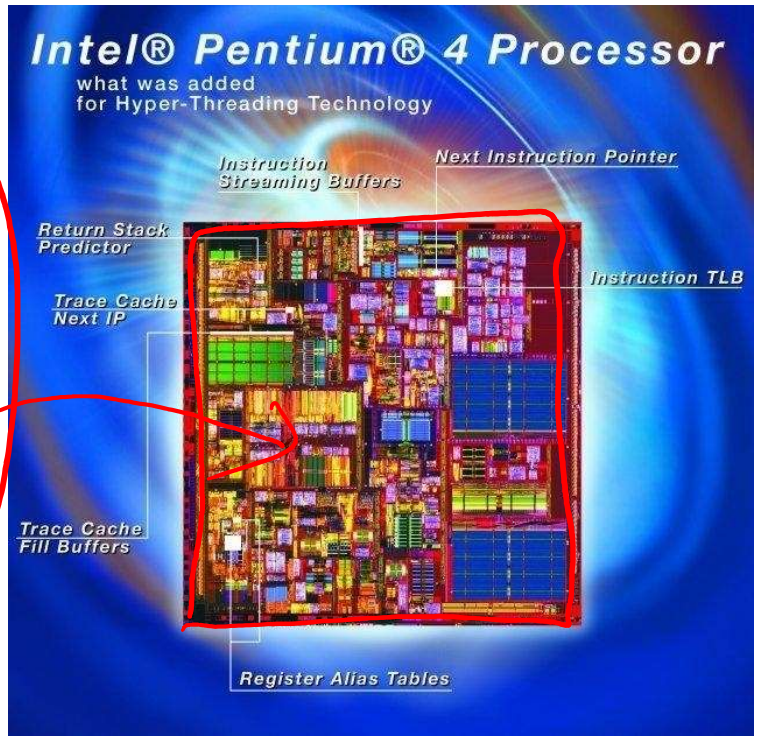
- organization of modern computers



Computer Organization

- microprocessor architecture





Intel Pentium 4 processor architecture block diagram and die photo

What to Expect from Taking ECE 411

- to understand fundamental concepts in computer architecture and how they impact application performance and energy efficiency
- to become confident in programming for performance, scalability, and efficiency
- to be able to understand and evaluate architectural descriptions of even today's most complex processors
- to gain experience designing a working CPU completely from scratch
- to learn experimental techniques used to evaluate advanced architectural concepts

Course Objective & Prerequisites

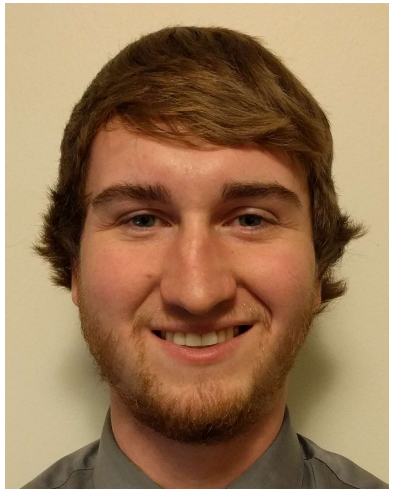
- fundamental concepts in computer organization focusing on uni-processor architecture
  - ✓ pipelining
  - ✓ memory hierarchy
  - ✓ instruction-level parallelism & dynamic scheduling
  - ✓ network & storage I/O subsystems
  - ✓ impact of technology on performance, energy efficiency and reliability
- prerequisites
  - ✓ ECE 391 or CS241 (ECE 385)
  - ✓ Linux
  - ✓ C++ programming
  - ✓ SystemVerilog
  - ✓ Quartus

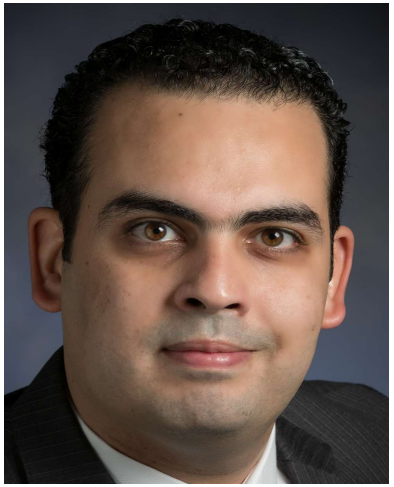
Course Information

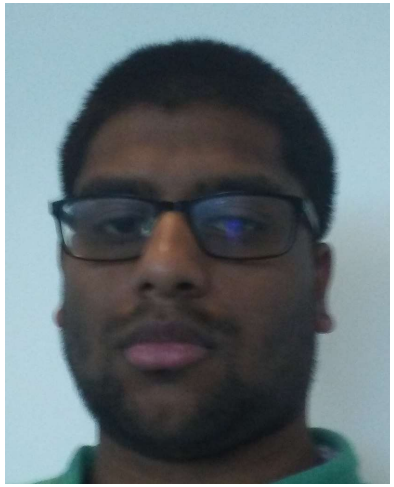
- instructor
  - ✓ Nam Sung Kim (nskim@Illinois.edu (the subject line w/ "[ECE411]"))
  - ✓ profile
    - Ph.D. from University of Michigan Ann Arbor (2004)
    - Sr. Research Scientist (CPU architect) at Intel (2004-2008)
    - Assistant/Associate Professor at UW-Madison (2008-2013)
    - Associate Professor at UIUC (2013-)
  - ✓ primary research area
    - energy-efficient computing
  - ✓ office
    - CSL 217
  - ✓ office hour
    - Wed 3-4pm

Course Information

- teaching assistants


Chance Coats


Ahmed Abulila

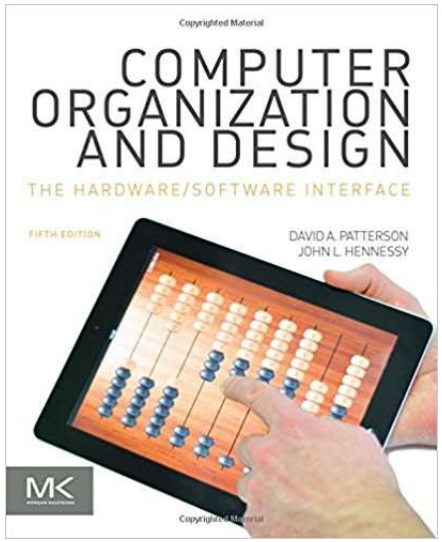

Krishna Srinivasan

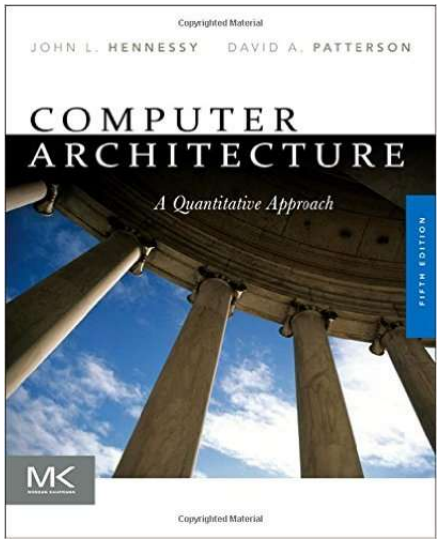

Kenneth Umenthum

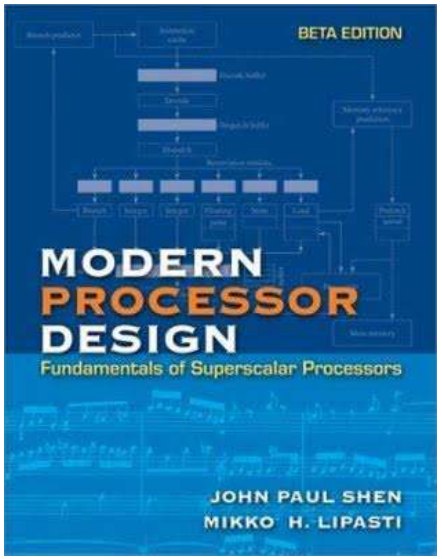
- contact information & office hour
  - ✓ Chance Coats (cccoats2@illinois.edu); 11am-1pm on Tue
  - ✓ Ahmed Abulila (abulila2@illinois.edu); 10-12pm on Mon
  - ✓ Krishna Srinivasan (kpsrini2@illinois.edu); 6-8pm on Thu
  - ✓ Kenneth Umenthum (umenthu2@illinois.edu); 1-3pm on Fri
- office hour location
  - ✓ DCL EWS Lab L520

Textbooks

- course textbook
  - [HP1] Patterson & Hennessy. Computer Organization and Design (5th Ed.); Morgan Kaufmann; ISBN: 978-0-12-407726-3
- recommended textbook
  - [HP2] Hennessy & Patterson. Computer Architecture: A Quantitative Approach (5th Ed.); The Morgan Kaufmann, ISBN: 978-0123838728
  - [SL] Shen & Lipasti. Modern Processor Design: Fundamentals of Superscalar Processors (1st Ed.); Waveland Press, ISBN: 978-1478607830







Lectures, MP & Assignments

- lectures
  - ✓ important to attend lecture faithfully as a notable fraction of materials are not explained in textbooks
  - ✓ a few "unannounced" in-class quizzes
    - ○ one quiz w/ the lowest score will not be counted for the final grading to accommodate an unforeseen event.
- MP assignments
  - ✓ there will be 4 MP assignments using SystemVerilog and Quartus
  - ✓ MP0, MP1, and MP2 done individually
  - ✓ MP3 done in teams of 3 (design project – 3 checkpoints)
  - ✓ 40% of final grade (5%+5%+10%+20%)
- HW assignments
  - ✓ there will be 2 HW assignments using a computer architecture simulator, SimpleScalar and CACTI, the use of which requires C/C++ programming skills

Exams & Grading

- exams
  - ✓ mid-term 1
    - 3/1 (15%)
  - ✓ mid-term 2
    - 4/10 (15%)
  - ✓ final
    - TBA (b/w 5/4 – 5/11
  - ✓ there will be a review lecture before each exam
- "curved" but…
  - ✓ in principle, everybody can make an "A+"
    - there will be some opportunities for extra credit primarily through MP3 and design competition
- late submission and missing quiz policies
  - ✓ each student will get 3 flexible days to use during MP0 through MP2. During a late submission, a student may specify in the electronic submission on Piazza how many flexible days they want to use. That is, in the private note simply state how many flexible days you would like to use.

Schedule

- syllabus

Website, Discussion Group & Staff Contact

- course website
  - ✓ lecture notes, handouts, MP assignment, etc.
  - ✓ announcements
- piazza
  - ✓ posting clarification questions
  - ✓ general forum to communicate with students, TAs, instructor about aspects of the course.
  - ✓ must join by the end of the day

Academic Integrity

- you are encouraged to discuss assignments w/ anyone you feel may be able to help you (except during exams)
- everything you turn in must be your work or that of your team
- we encourage collaboration
  - ✓ verbal discussion of problems/solutions
  - ✓ diagrams, notes, other communication aids
- unacceptable
  - ✓ copying/exchanging of program/verilog code or text by any means
  - ✓ giving/receiving help on exams, or using any notes/aid beyond those explicitly permitted

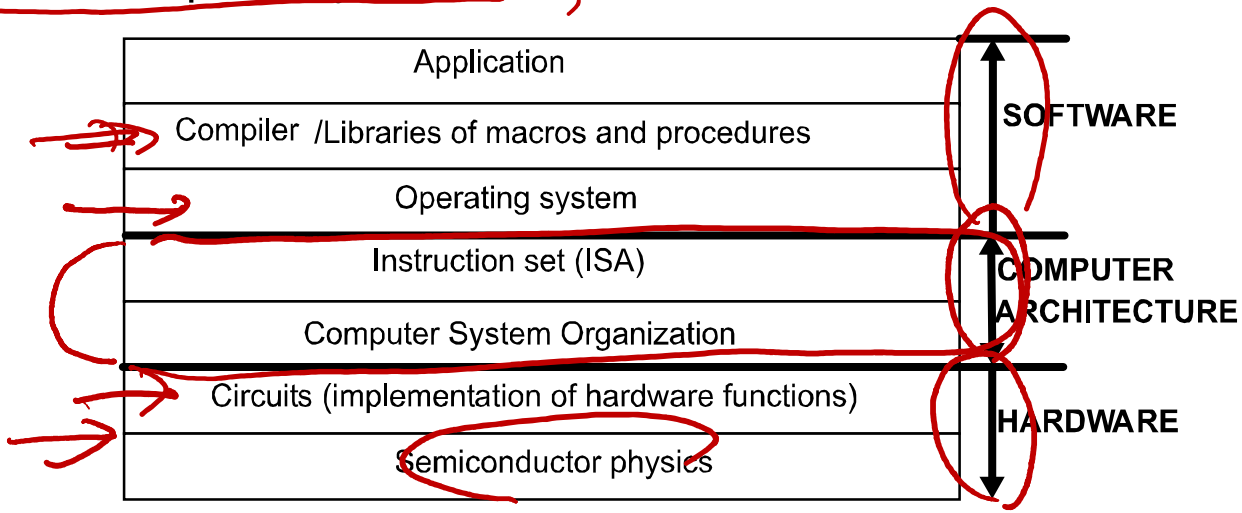
This course is not for everyone …

Requires a lot of time commitment

What is computer architecture and why does it matter?

What Is Computer Architecture?

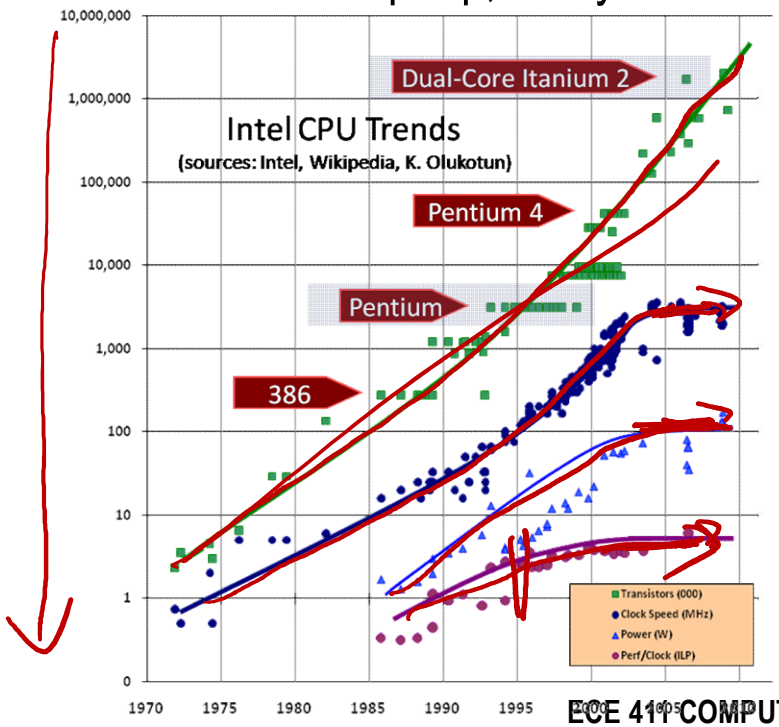
- old definition
  - ✓ developing instruction set architecture (ISA)
- today's definition is much broader
  - ✓ HW organization of computers including ISA
- layered view of computer systems

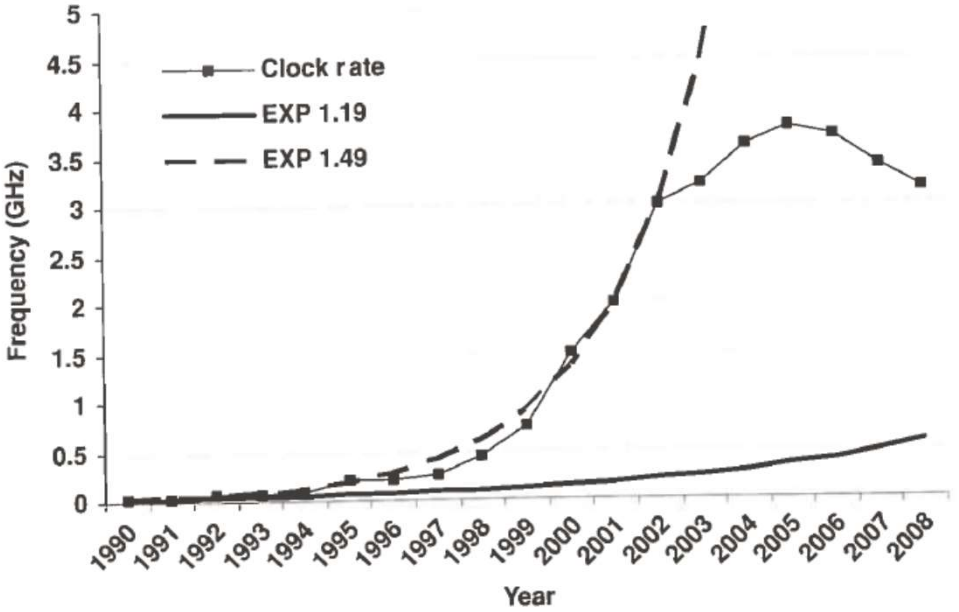


- role of computer architects
  - ✓ to make design trade-offs across HW/SW interface to satisfy functional, performance, power/energy, and cost requirements

Processor Architecture

- historically the clock rates have increased exponentially
  - ✓ highest clock rate of Intel processors from 1990 to 2008
    - ○ due to semiconductor technology improvements
    - ○ deeper pipeline
    - ○ circuit design techniques
- this historical trend has subsided over the past 10 years
  - ✓ If it had kept up, today's clock rates would be more than 30GHz





Power Efficiency Challenges

ASCI Red @ Sandia Labs

1997 (850KW) **Teraflops** **201x (5W)**

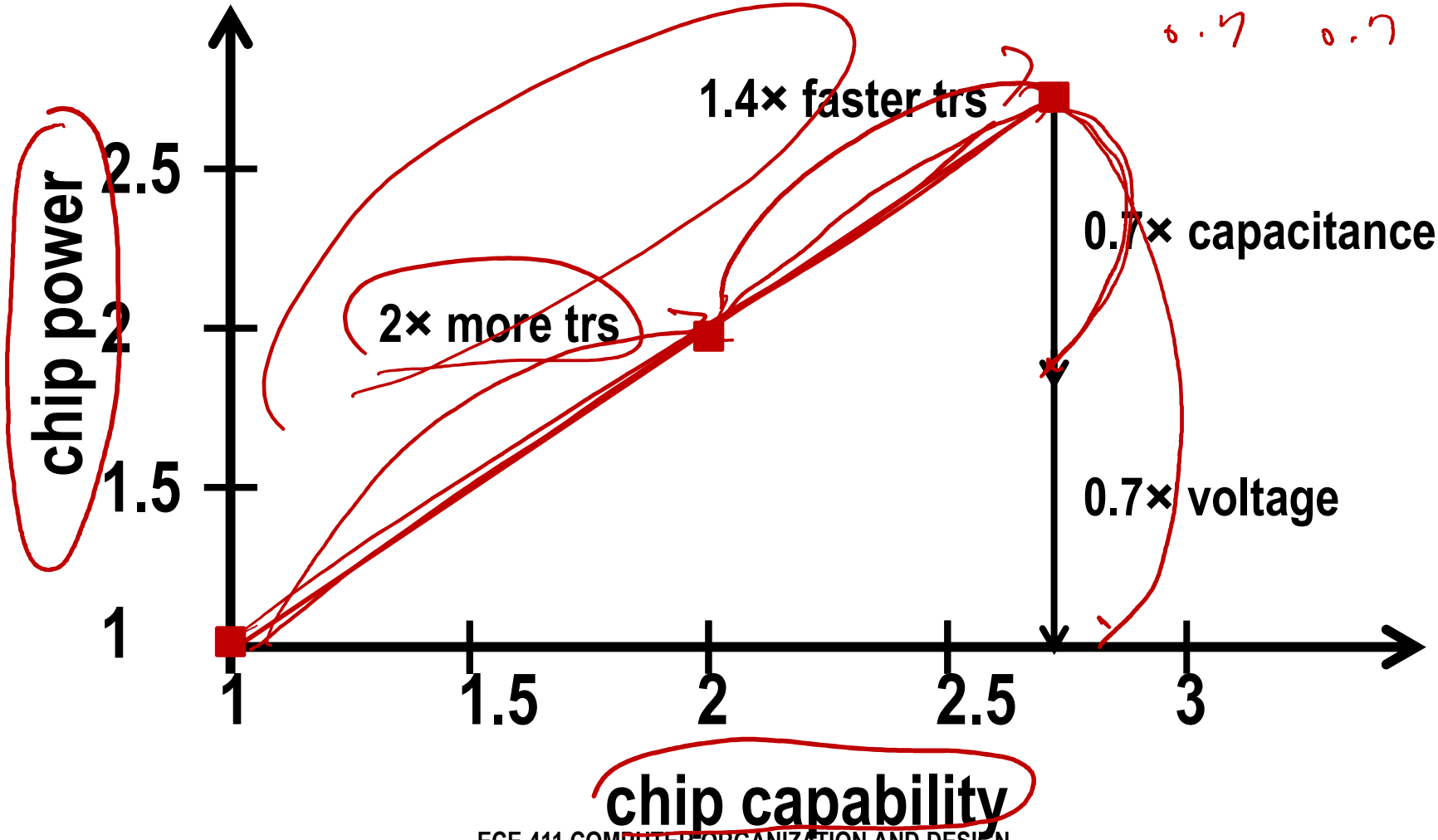
- power/energy-constrained computing devices
  - ✓ mobile: 1~5W power budget w/ 5Whr battery capacity

Classic Dennard's Scaling

- 0.7× transistor feature size scaling per generation
  - ✓ 2.8× more chip capability at the same power

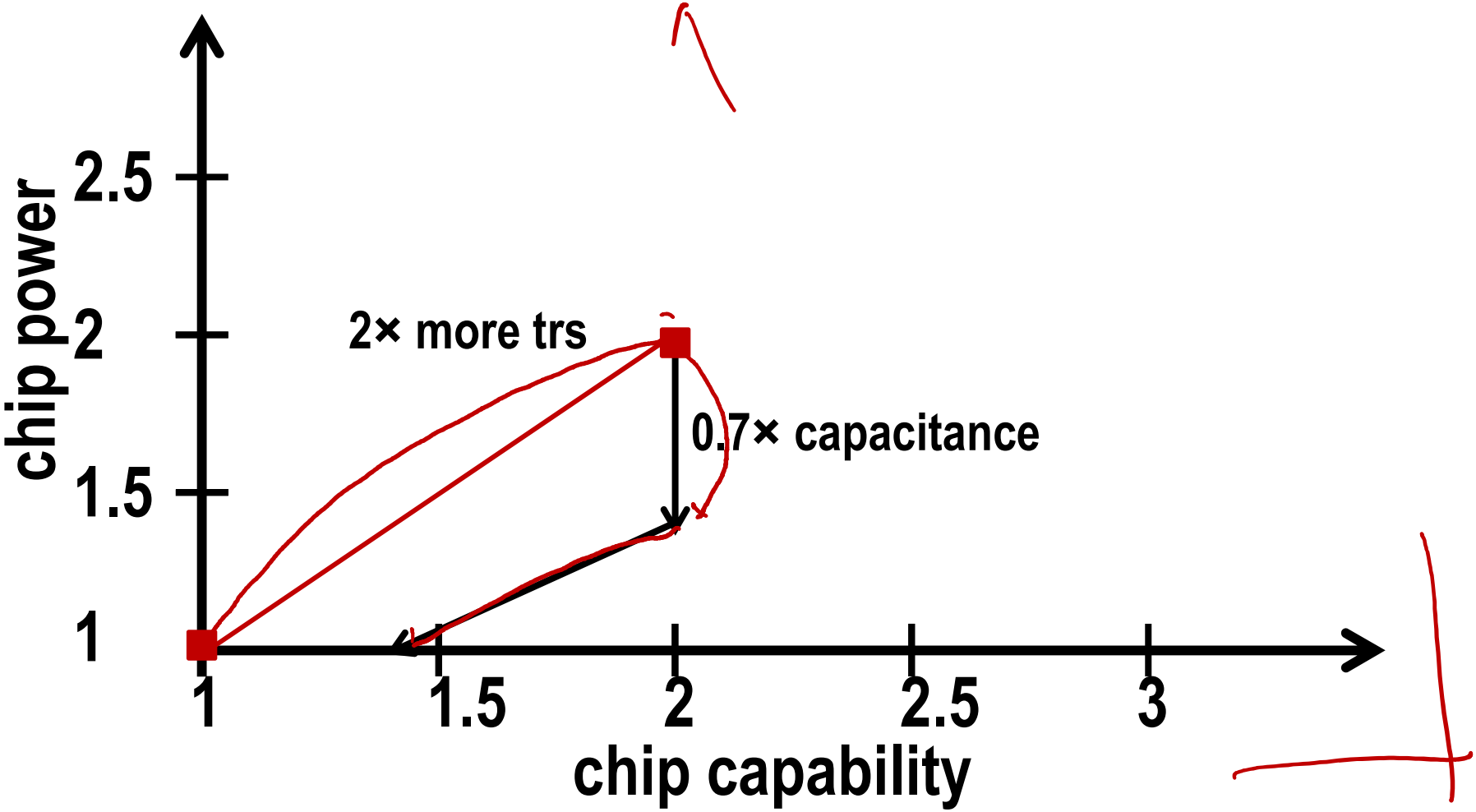
$P = \xi V^2 f$

$0.7 \quad 0.7 \quad 2$

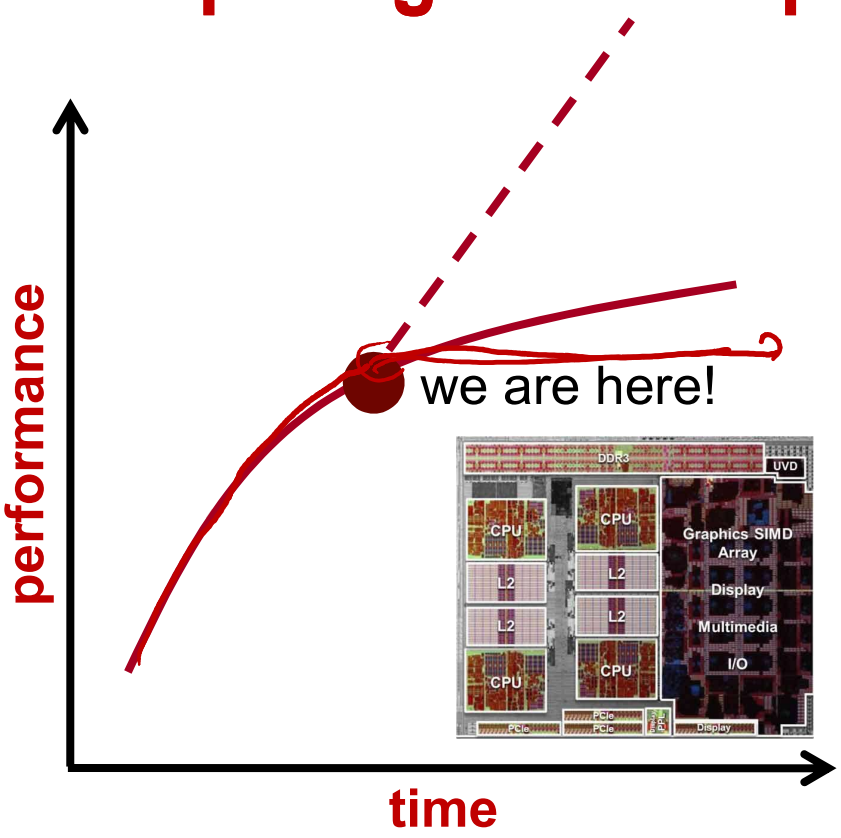


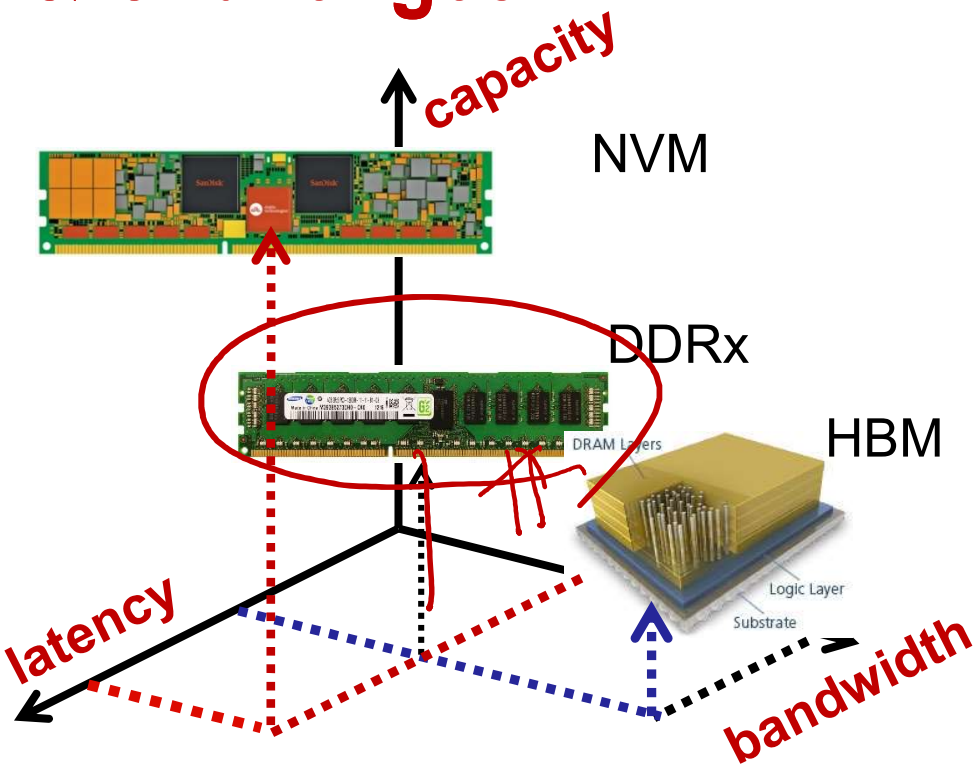
End of Dennard's Scaling

- 0.7× transistor feature size scaling per generation
  - ✓ 1.4× more chip capability at the same power

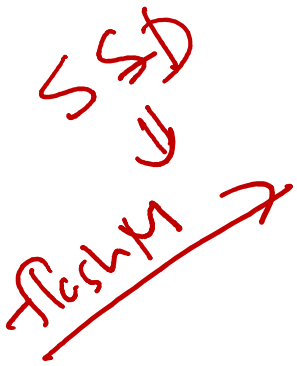


Computing Landscape & Challenges



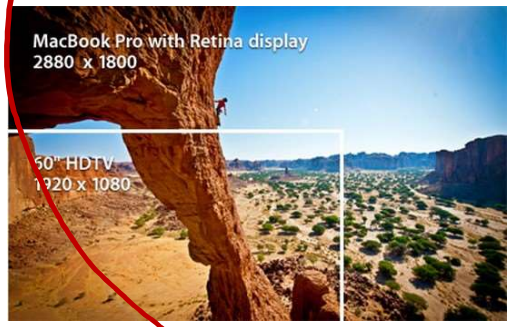


- diminishing performance of **general-purpose CPU-based** system
  - ✓ **power constraint**
    - acceleration using CPU+**GPU (GPGPU computing)**
  - ✓ **memory capacity and bandwidth constraints**
    - capacity using NVM
    - bandwidth using 2.5D/3D-integrated **HBM**

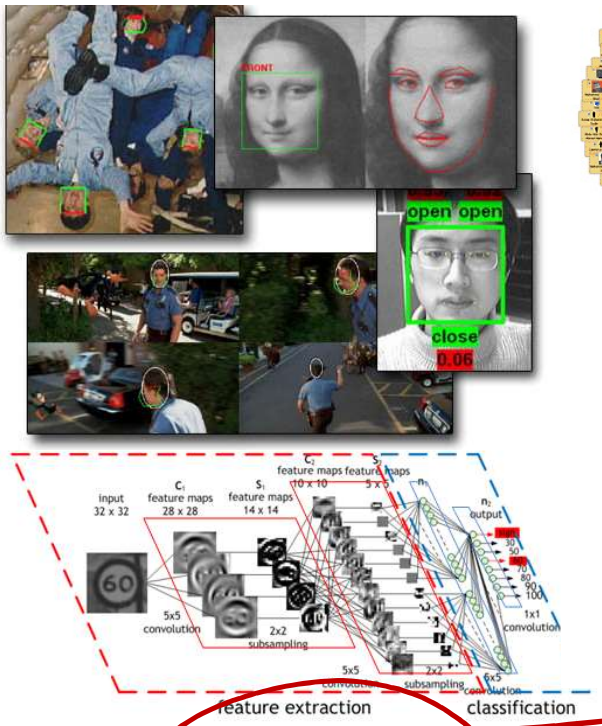


Emerging Apps & Characteristics

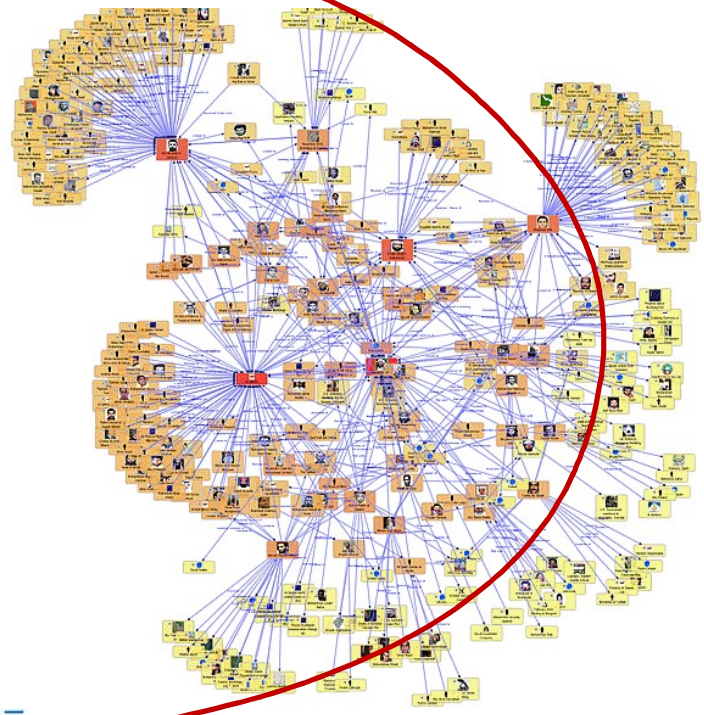
scientific/engineering



SUHD/3D-graphics



recognition/mining
(machine-learning apps)

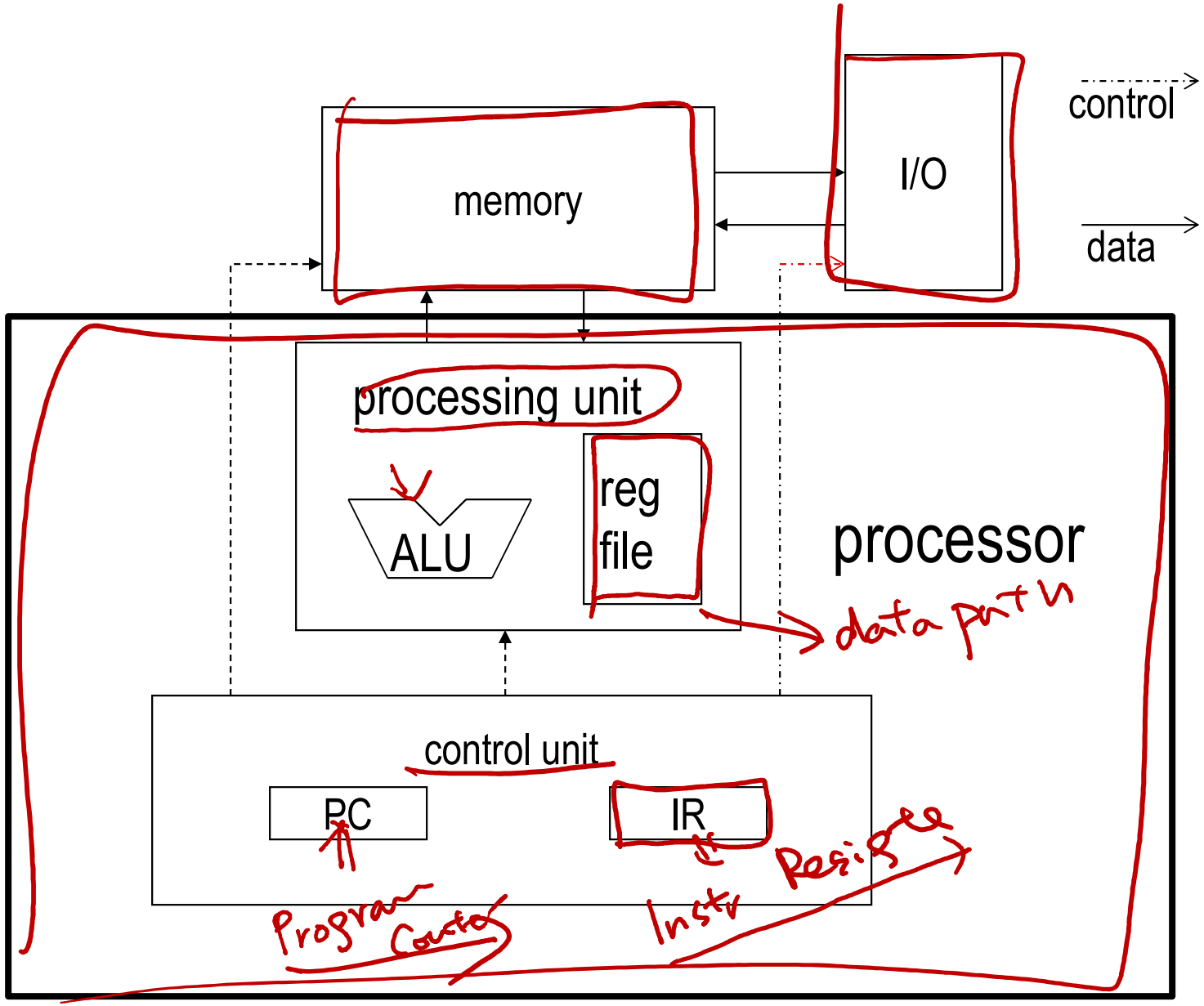


DB/SNS/websearch
(datacenter/big-data apps)

traditional apps	emerging apps
✓ complex/ irregular ops	✓ simple repetitive ops
✓ large # of ops per value	✓ small # of ops per value
✓ high temporal locality	✓ low-temporal locality

Von-Neumann computer architecture

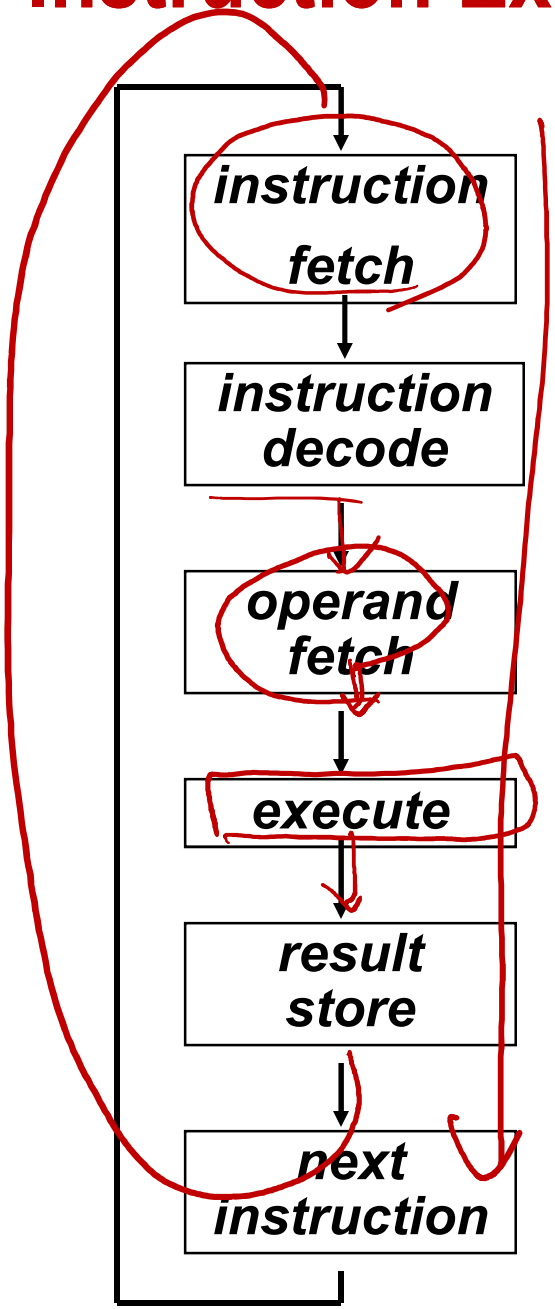
Von-Neumann Computer Organization



Von-Neumann Computer Organization

- executes instructions, which are represented as bit patterns with specific fields interpreted differently
- contains a register file, which holds data that will be referenced by instructions
- has a program counter that holds the address of the instruction being executed
- can access memory
  - ✓ use address to select the location we want to access
  - ✓ random-access: time to access a piece of data is independent of which address the data is stored in
- I/O
  - ✓ keyboard, mouse, display, network
  - ✓ long-term data storage: hard disk, CD-ROM, SSD, etc.

Instruction Execution Cycles



main memory

obtain instruction from program storage

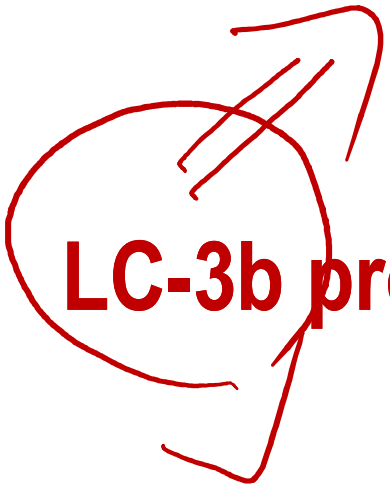
determine required actions and instruction size

locate and obtain operand data

compute result value or status

deposit results in storage for later use

determine successor instruction

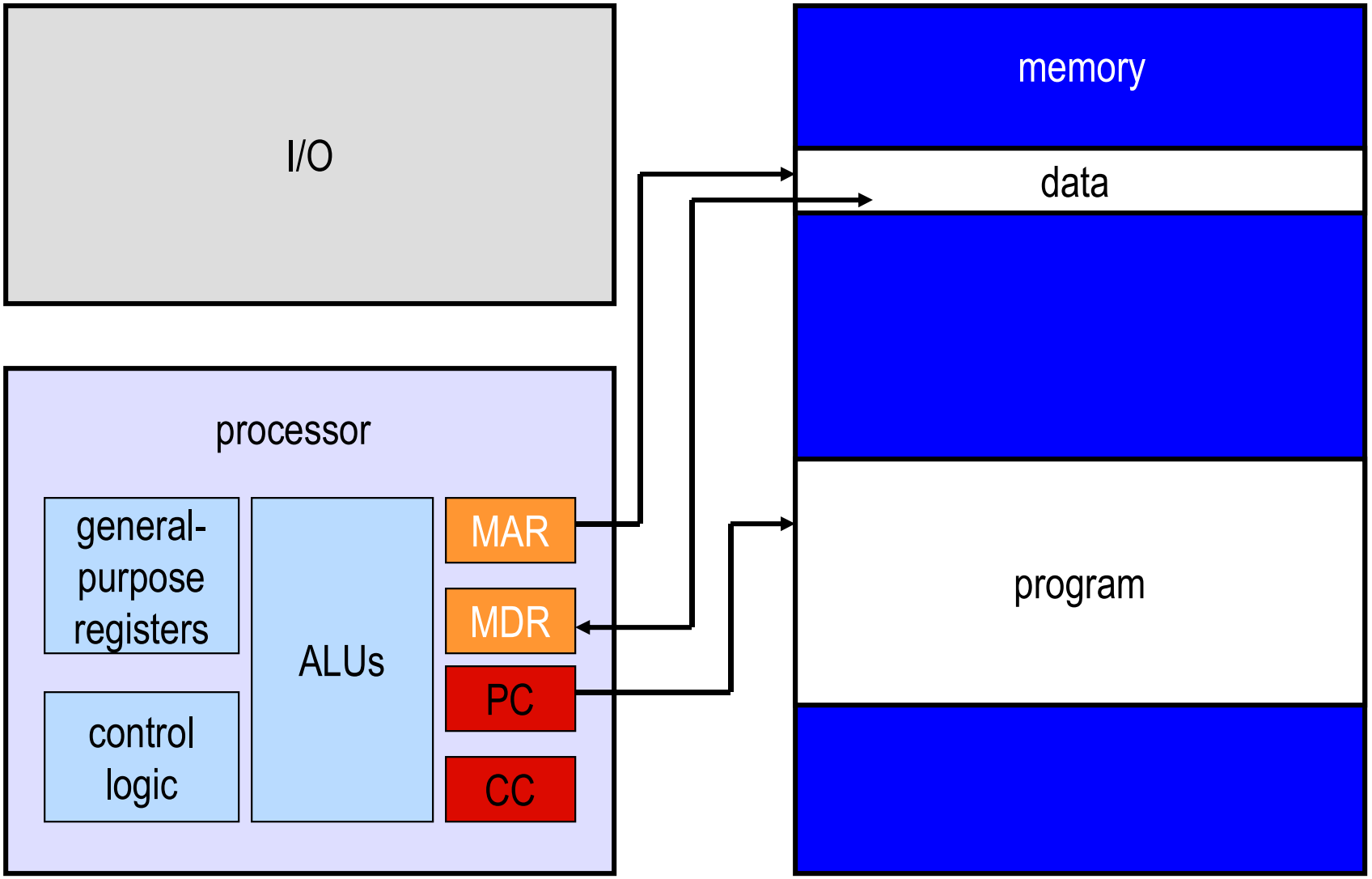


LC-3b processor architecture

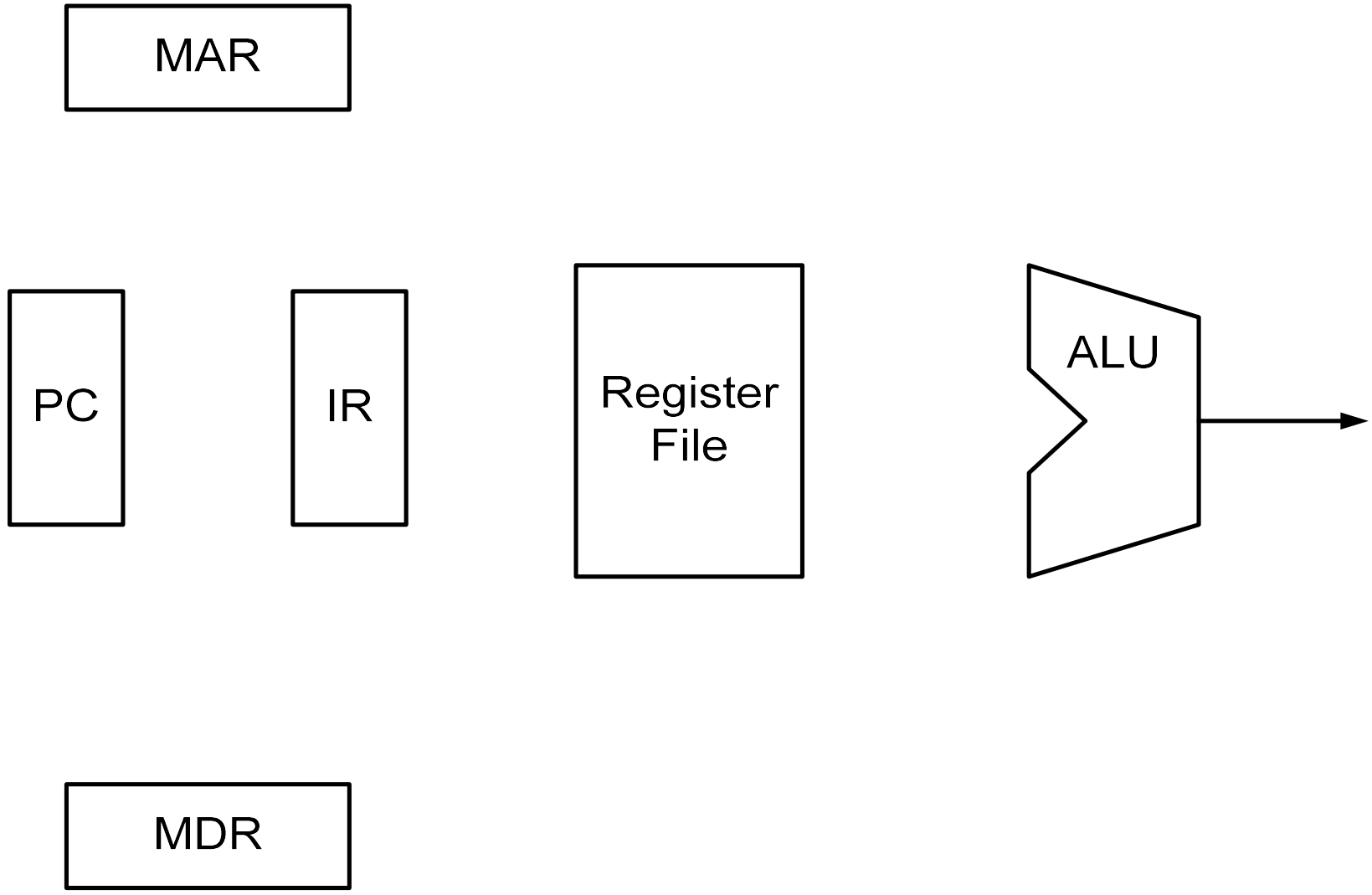
LC-3b Instruction Set Encodings

- instructions represented as 16-bit words
- opcode always high four bits of the word
  - ✓ this is one case where the LC-3b is much more simplistic than commercial ISAs, which generally have several different opcode formats for different types of instructions
- a few instruction formats
  - ✓ format
    - ○ mapping of bits in the instruction to operands/outputs/etc.
  - ✓ commercial ISAs often have many more formats

Basic Computer Organization – more details



Datapath: ALUs Plus Registers



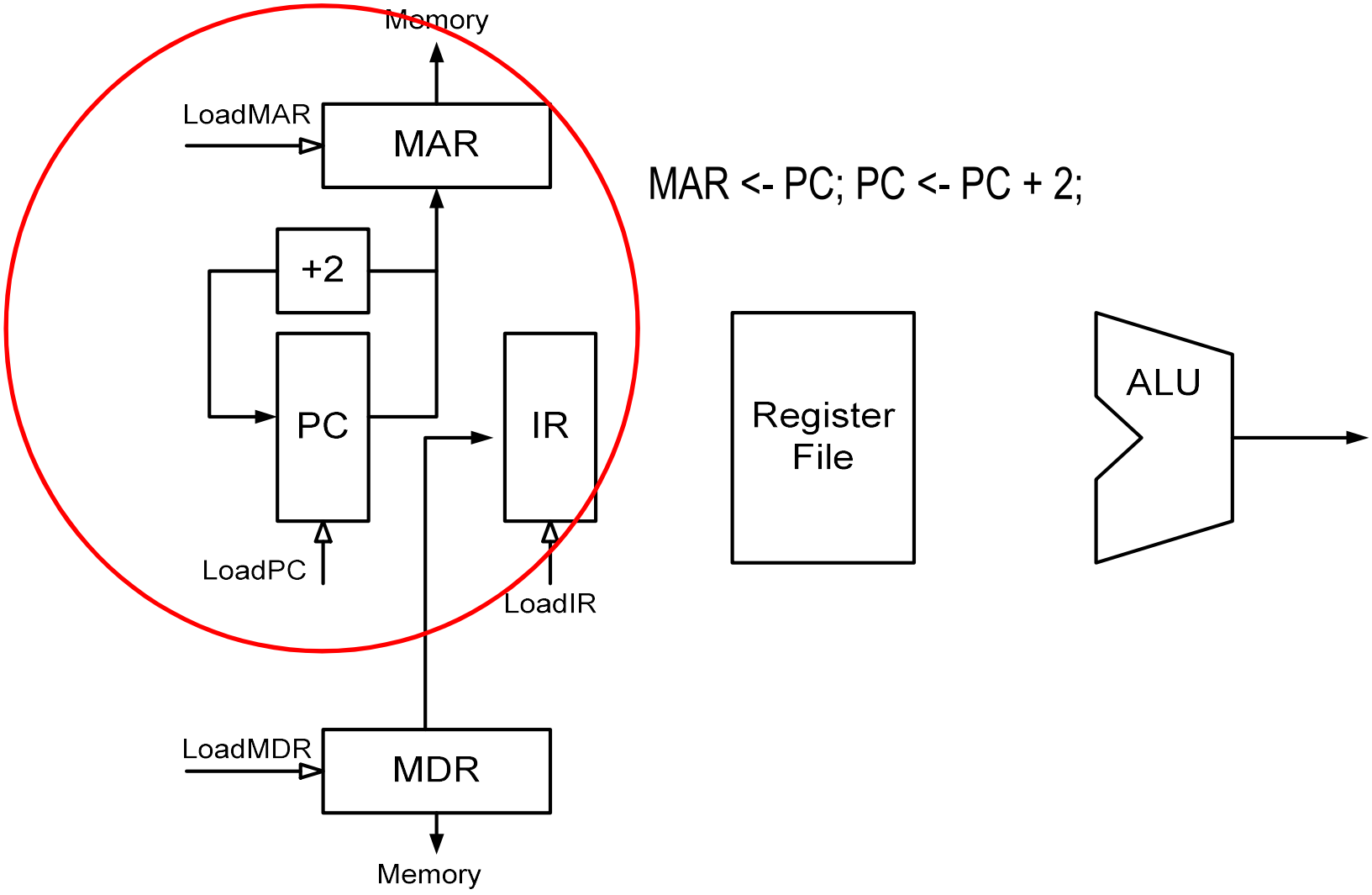
Register Transfer Level Notation for Fetch

- one line per clock cycle
- tells you how data moves b/w registers
- can have multiple operations in parallel

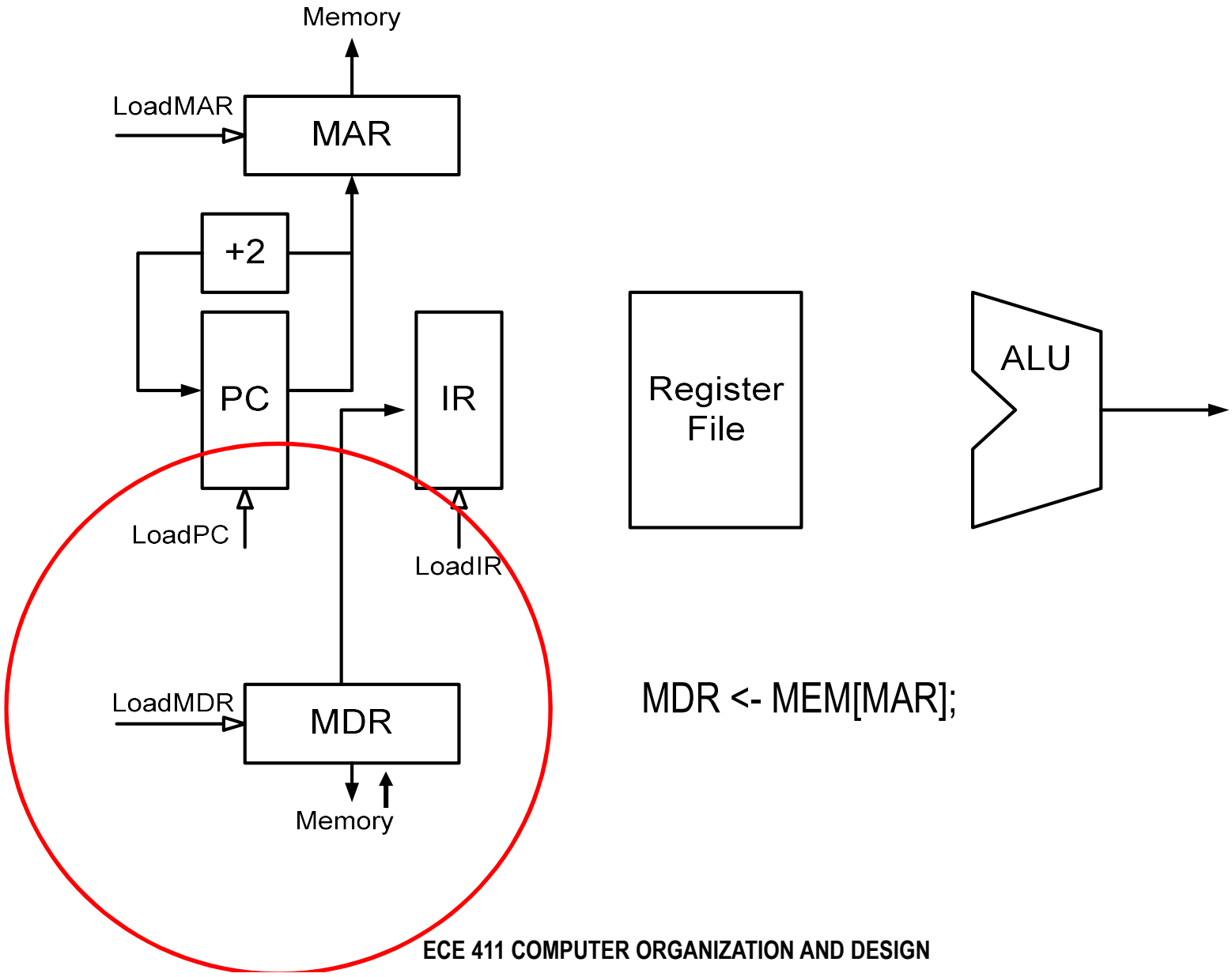
RTL for the instruction fetch:

```
MAR <- PC; PC <- PC + 2;
MDR <- MEM[MAR];
IR <- MDR;
```

Fetching Instructions

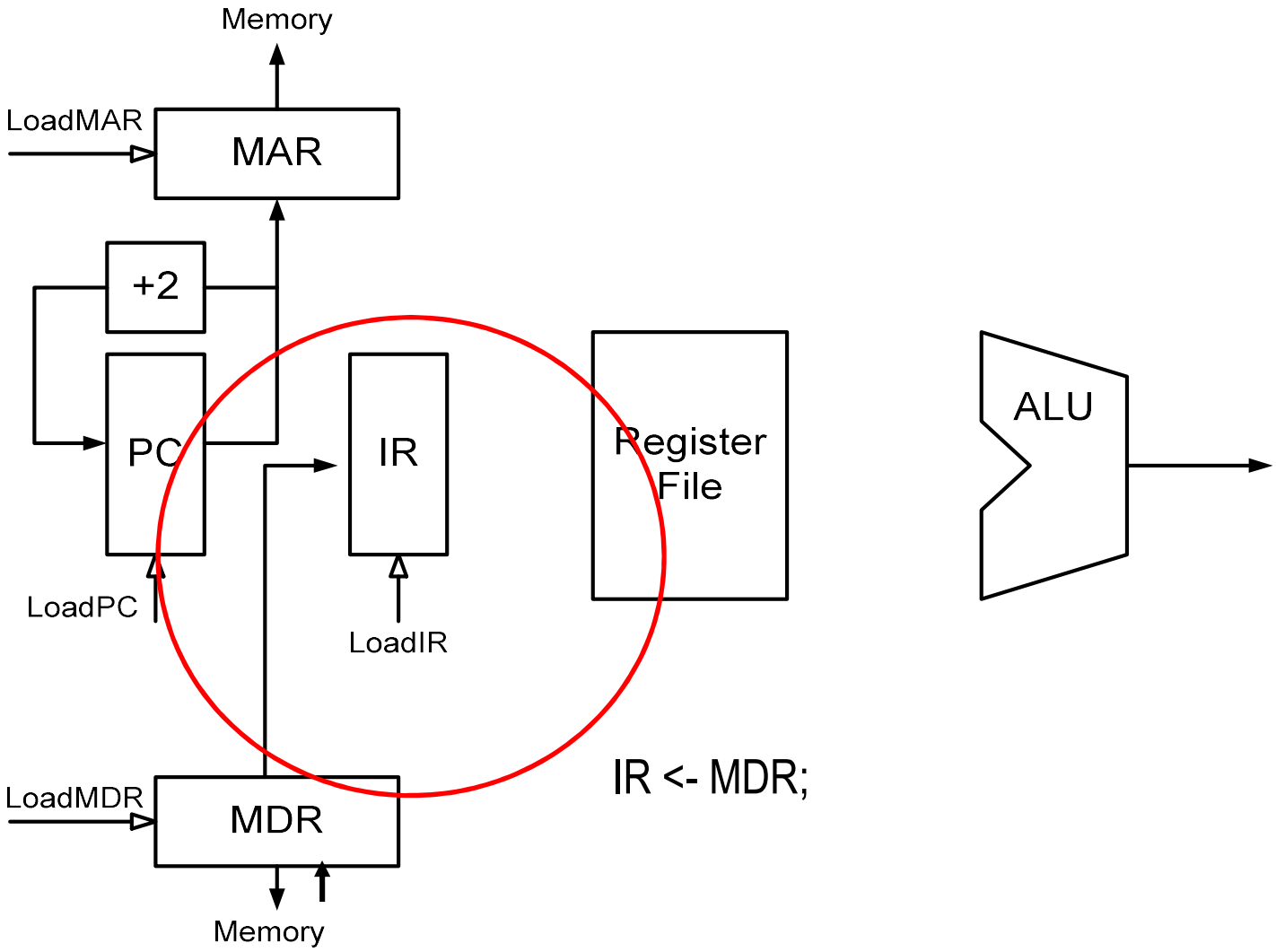


Fetching Instructions



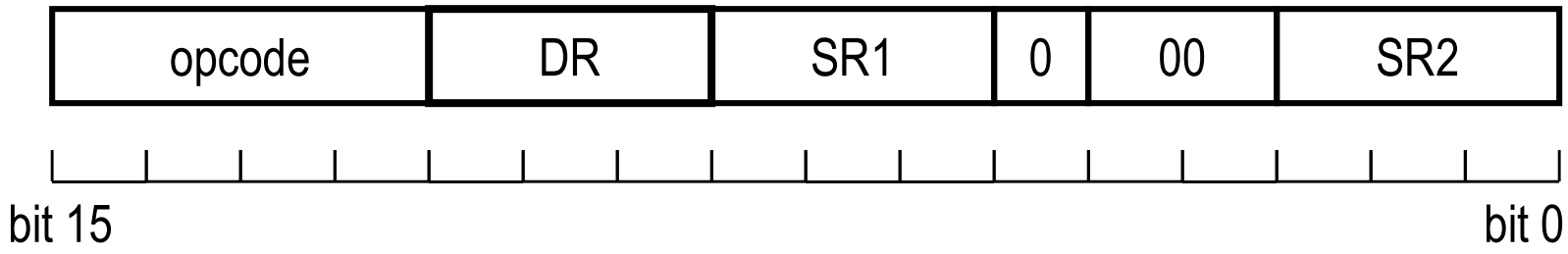
MDR <- MEM[MAR];

Fetching Instructions

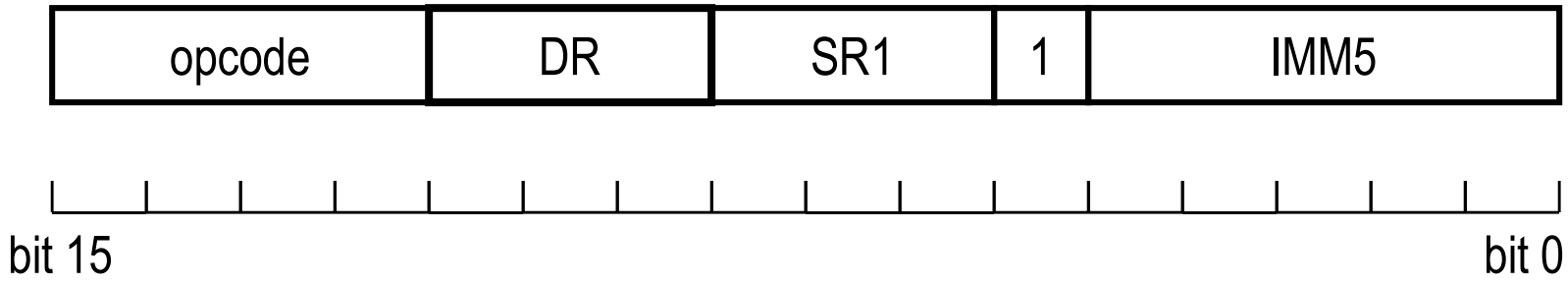


LC-3b Instruction Encoding Examples

- ADD, AND (w/o immediate)



- ADD, AND (w/ immediate), NOT



RTL For ADD/AND Instructions

w/o immediate

MAR <- PC; PC <- PC + 2;

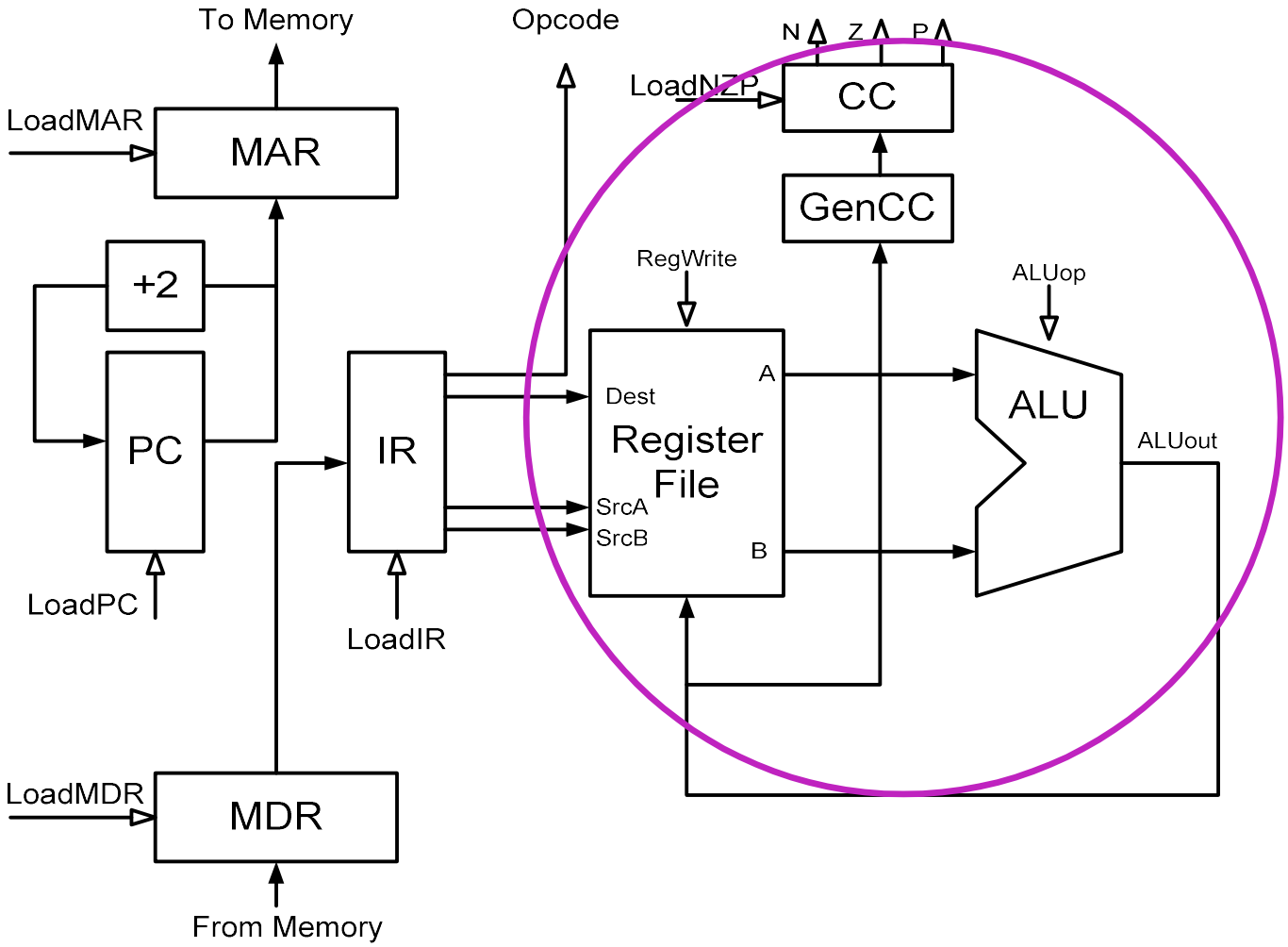
MDR <- MEM[MAR];

IR <- MDR;

R[DR] <- R[SR1] OP R[SR2]; genCC;

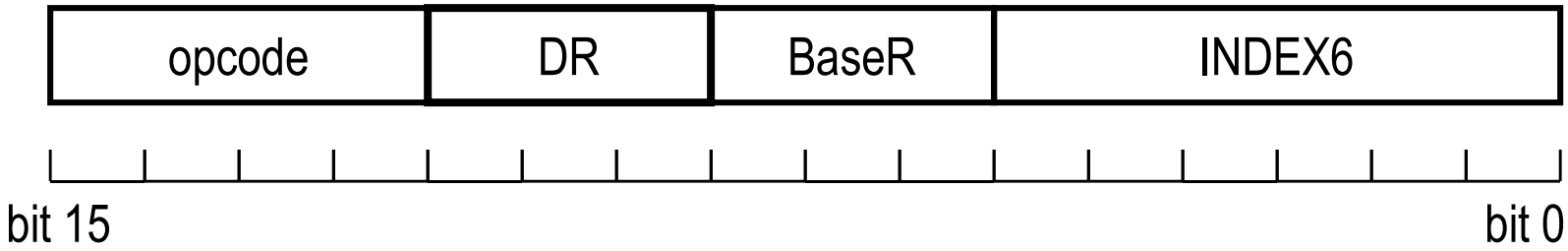
Arithmetic Instructions: AND, ADD, NOT

R[DR] <- R[SR1] OP R[SR2]; genCC;



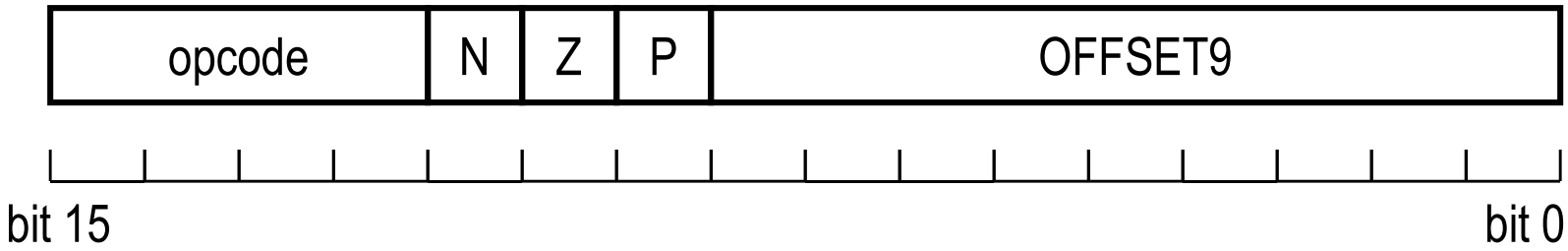
LC-3b Instruction Encoding Examples

- LD, LDI, LDB



LDB R4, R2, #-5; R4 <- mem[R2-5]

- BR



RTL For LD, ST

LD

MAR <- PC; PC <- PC + 2;

MDR <- MEM[MAR];

IR <- MDR;

MAR <- R[IR[8:6]] + ADJ6(IR[5:0]);

MDR <- MEM[MAR];

R[IR[11:9] <- MDR; genCC;

ST

MAR <- PC; PC <- PC + 2;

MDR <- MEM[MAR];

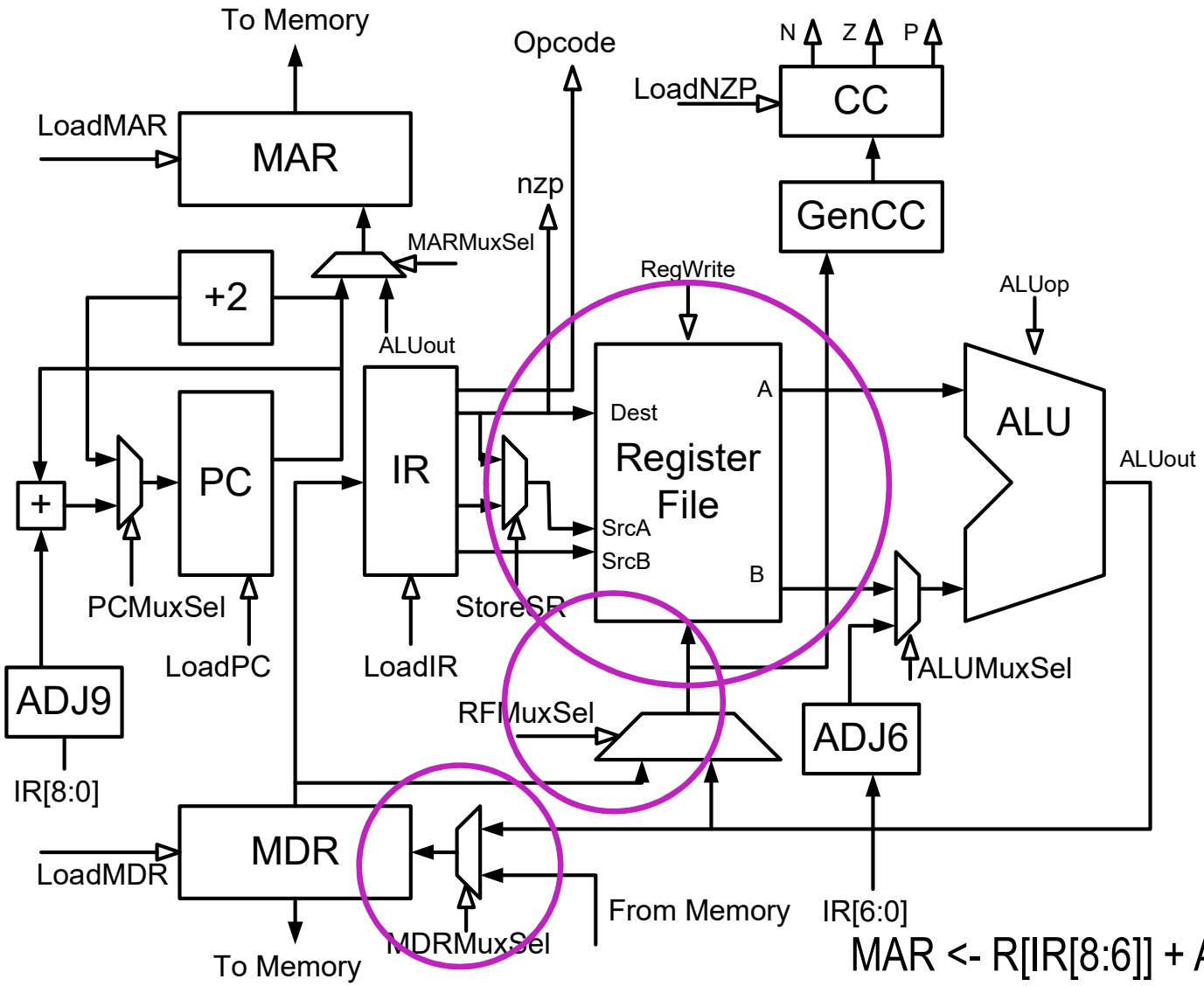
IR <- MDR;

MAR <- R[IR[8:6]] + ADJ6(IR[5:0]);

MDR <- R[IR[11:9];

MEM[MAR] <- MDR;

Refined Data Path for Memory: LD

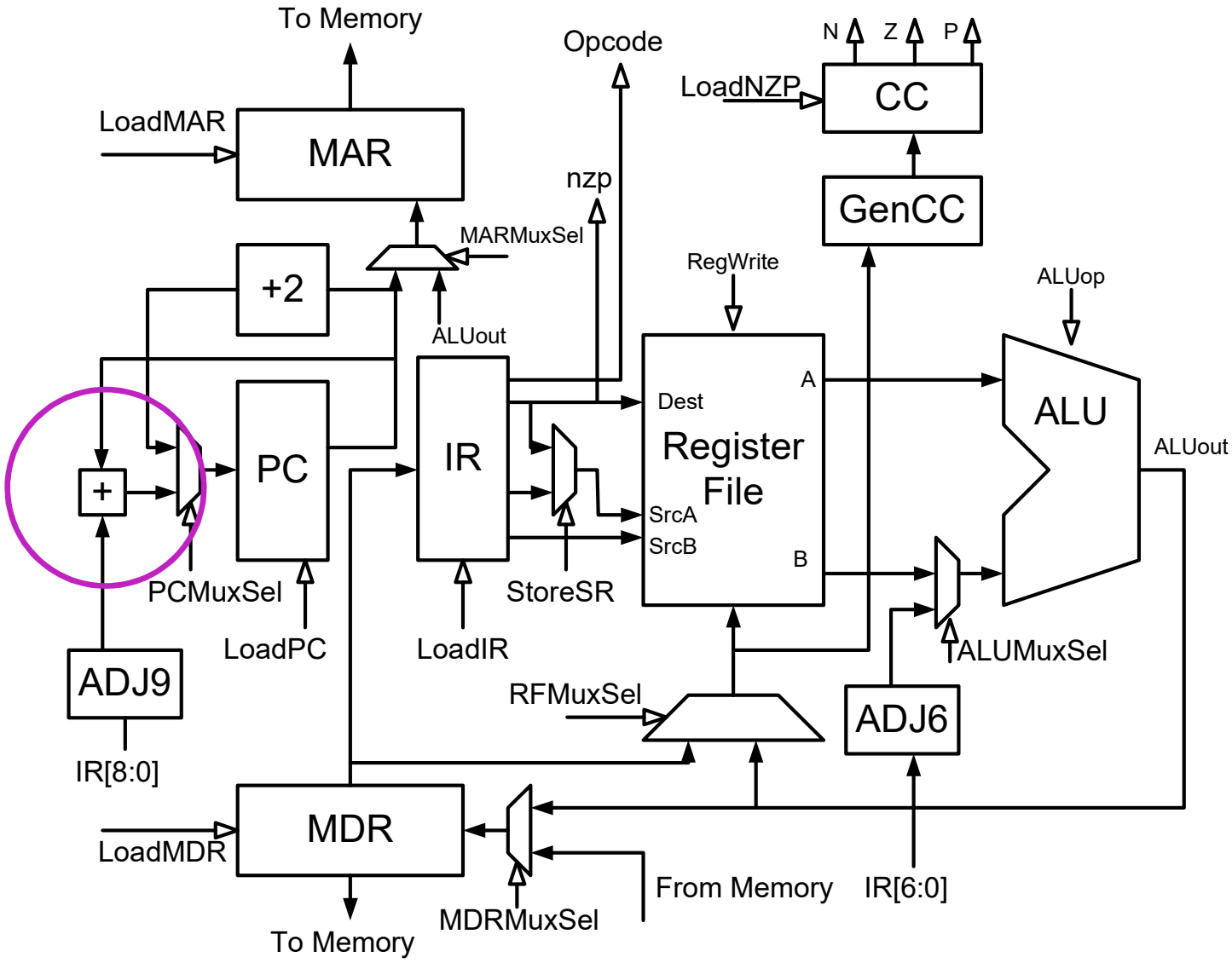


MAR <- R[IR[8:6]] + ADJ6(IR[5:0]);
MDR <- MEM[MAR];
R[11:9] <- MDR; genCC;

RTL for BR

```
MAR <- PC; PC <- PC + 2;
MDR <- MEM[MAR];
IR <- MDR;
If ((n AND N) OR (z AND Z) OR (p AND P))
    PC <- PC + ADJ9(IR[8:0]);
```

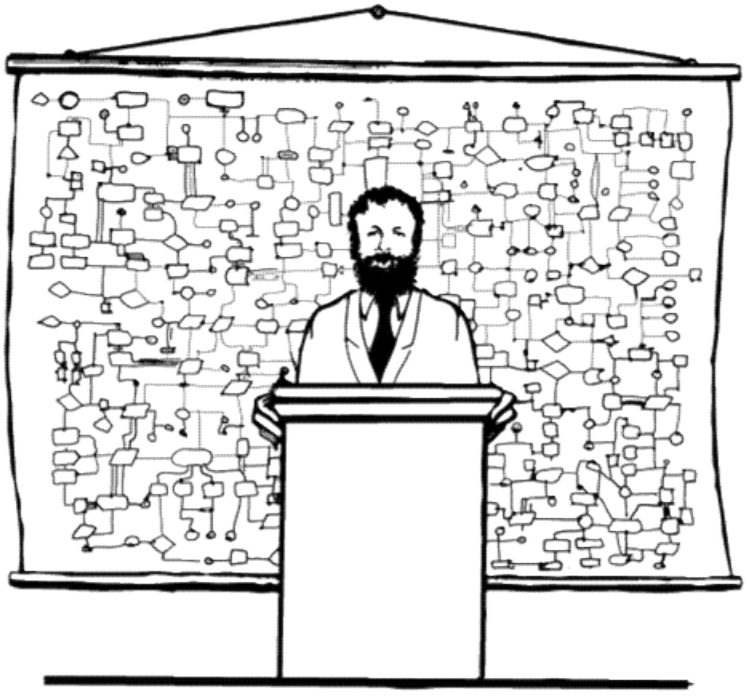
Refined Data Path for Control: BR



Announcement

- next lecture
  - ✓ instruction set architecture
    - [HP2] Appendix A
- MP assignment
  - ✓ a tutorial
  - ✓ assignment available at ECE411 website
  - ✓ need to use EW Lab
- a bit more detail on LC3b

http://pages.cs.wisc.edu/~arch/www/cartoon.jpg



"Now that you have an overview of the system, we're ready for a little more detail"

https://qph.ec.quoracdn.net/main-qimg-790e56af154d4afc43ffd506a6215ce9