McMaster University | BRIGHTER WORLD

COMPENG 2SH4 Lecture 0

Introduction to Computer Programming

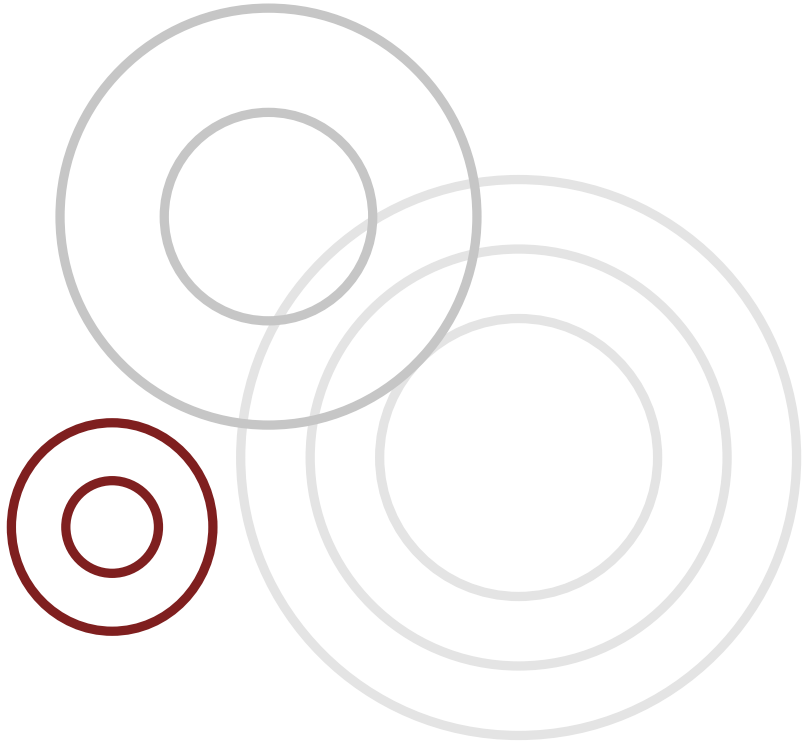
Dr. Shahrukh Athar

Credit: Dr. M. Hassan, 2SH4 F21; Dr. S. Chen, 2SH4 F22, F23

Agenda

- **Introduction to Computer Engineering**

- **Housekeeping**
  - Teaching Team
  - Instructional Plan
  - Setting the bars



About Me

- **Shahrukh Athar** athars3@mcmaster.ca
  - Ph.D. Electrical and Computer Engineering — University of Waterloo (2020)
  - Certificate in University Teaching — University of Waterloo (2020)

- **Experience**
  - Assistant Professor (teaching-focused), ECE Department, McMaster University (since August 2022)
  - University of Waterloo, Canada
    - Postdoctoral Fellow (2020-22)
    - Sessional Lecturer (W2020)
    - Graduate Instructional Developer (2017-18)
    - Graduate Teaching Assistant (2014-19)
  - LUMS, Pakistan: Lecturer, EE Department (2010-13)

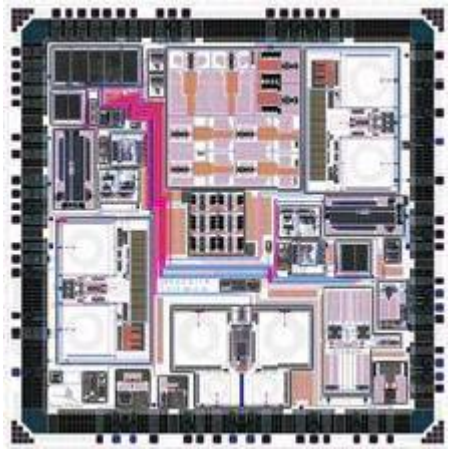
- **Research Interests**
  - Perceptual image quality assessment (IQA)
  - Degraded-reference (DR) IQA
  - Large-scale dataset design for IQA
  - Scholarship of teaching and learning

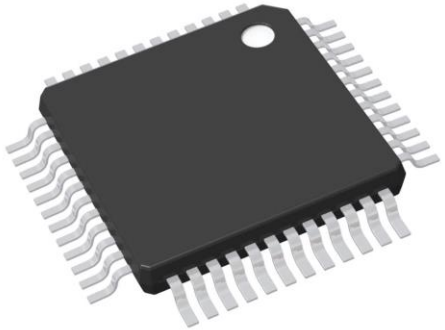
Introduction to Computer Engineering

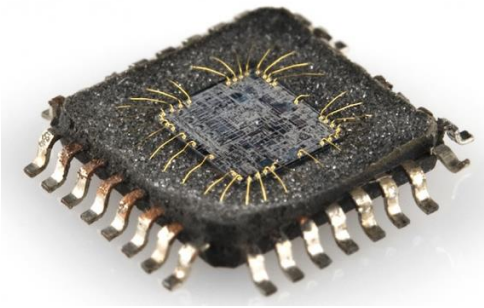
- How 2SH4 fits into your COMPENG program

Computer System – Full System Stack

- General Model of a Computer System



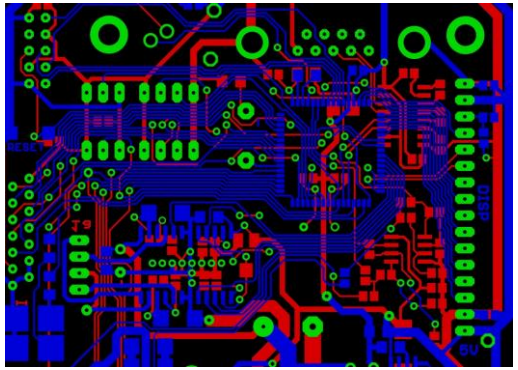


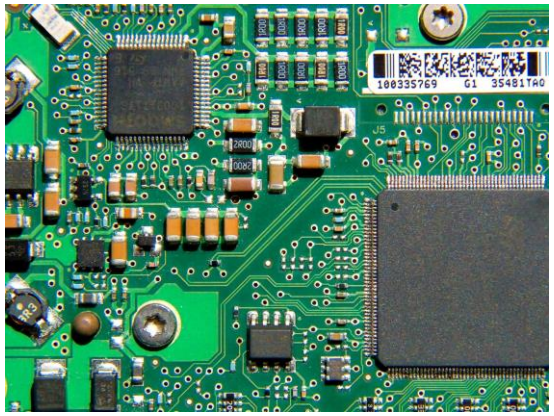


Physical (IC Layout) Design

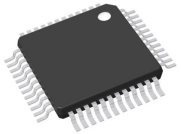
Computer System – Full System Stack

- General Model of a Computer System



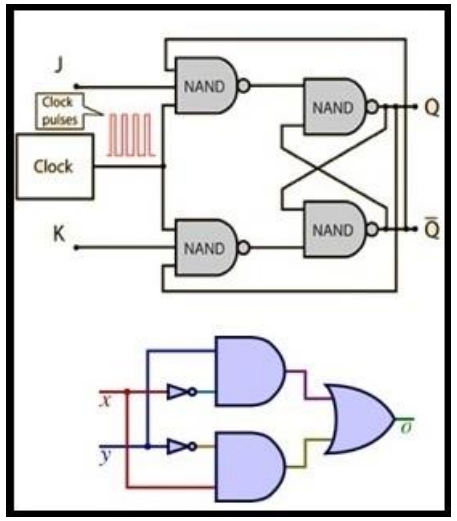


Circuit Design
Physical (IC Layout) Design

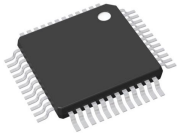


Computer System – Full System Stack

- General Model of a Computer System

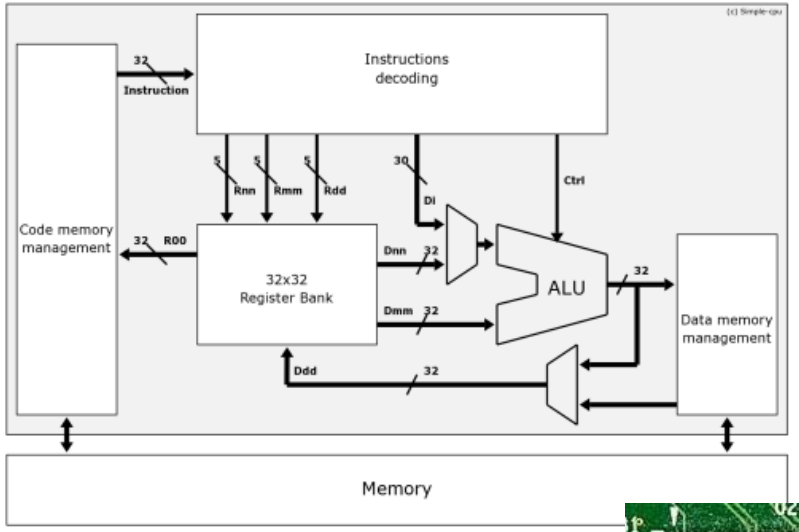


Digital Logic Design
Circuit Design
Physical (IC Layout) Design

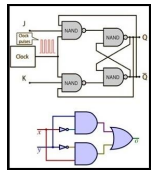


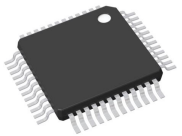
Computer System – Full System Stack

- General Model of a Computer System



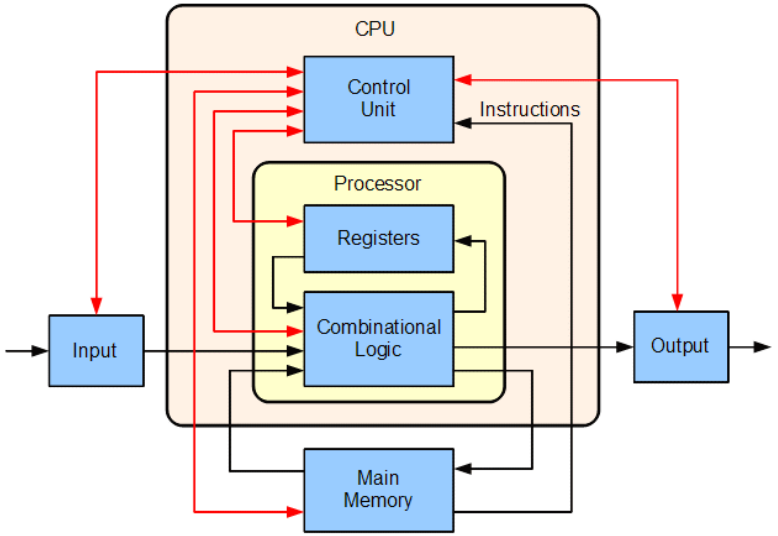
Datapath and Control Design
Digital Logic Design
Circuit Design
Physical (IC Layout) Design



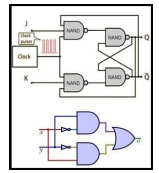


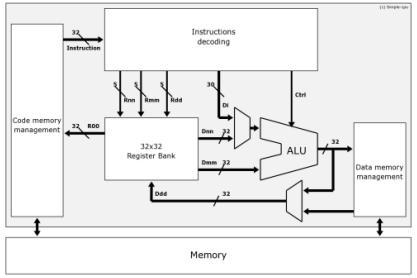
Computer System – Full System Stack

- General Model of a Computer System



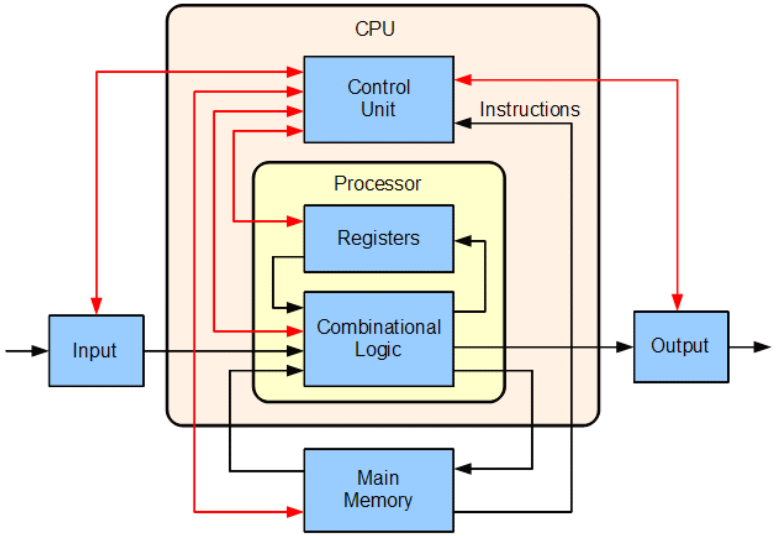
Processor	Memory	IO System
Datapath and Control Design		
Digital Logic Design		
Circuit Design		
Physical (IC Layout) Design		





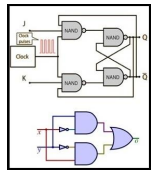
Computer System – Full System Stack

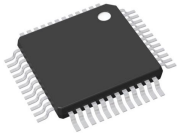
- General Model of a Computer System

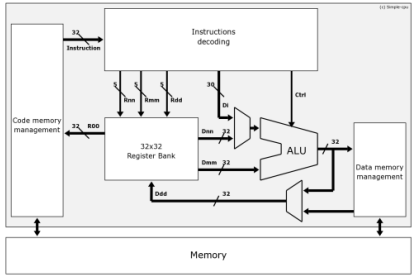


Hardware

Processor	Memory	IO System
Datapath and Control Design		
Digital Logic Design		
Circuit Design		
Physical (IC Layout) Design		



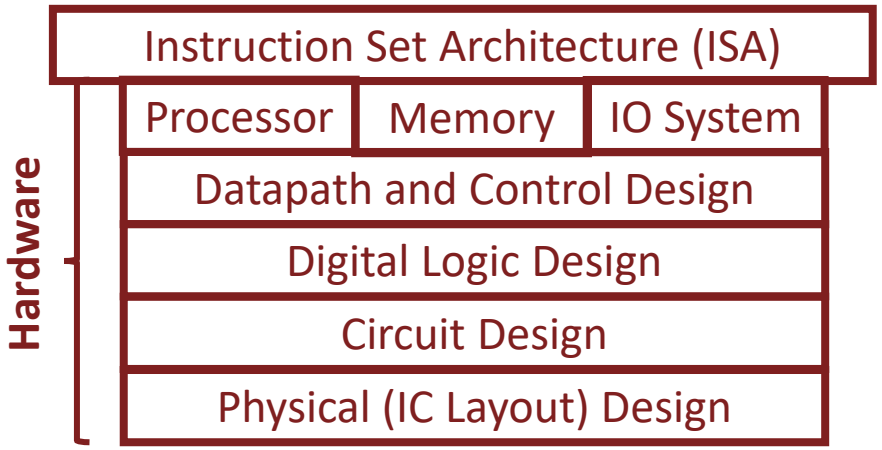


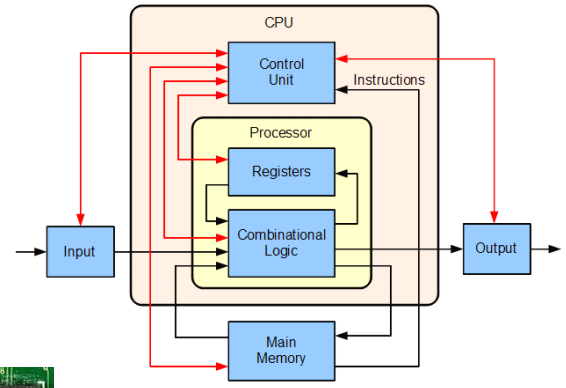


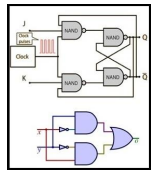
Computer System – Full System Stack

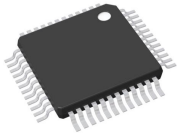
- General Model of a Computer System

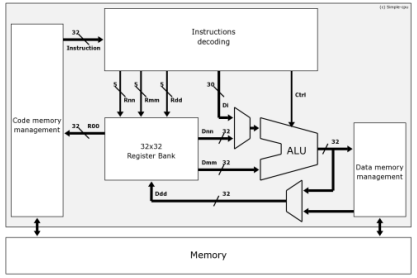
Instruction	Instruction Code	Operation
NOP	0000 0000	No operation
LDAC	0000 0001 Γ	$AC \leftarrow M[\Gamma]$
STAC	0000 0010 Γ	$M[\Gamma] \leftarrow AC$
MVAC	0000 0011	$R \leftarrow AC$
MOVR	0000 0100	$AC \leftarrow R$
JUMP	0000 0101 Γ	GOTO Γ









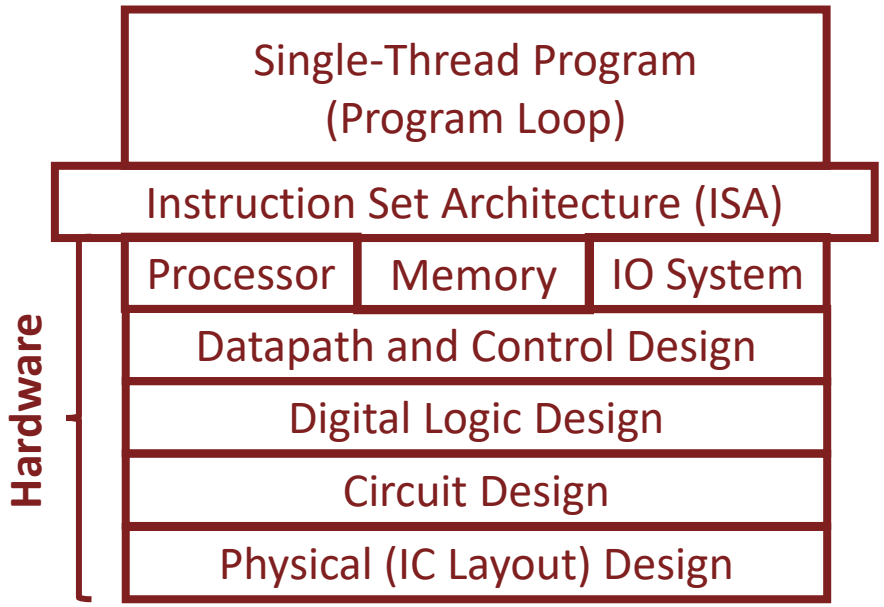


Computer System – Full System Stack

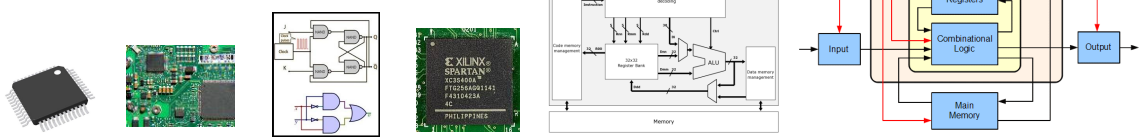
- General Model of a Computer System

```
2  int square(int num) {
3      return num * num;
4  }
```

```
1  square:
2          push    rbp
3          mov     rbp, rsp
4          mov     DWORD PTR [rbp-4], edi
5          mov     eax, DWORD PTR [rbp-4]
6          imul    eax, eax
7          pop     rbp
8          ret
```

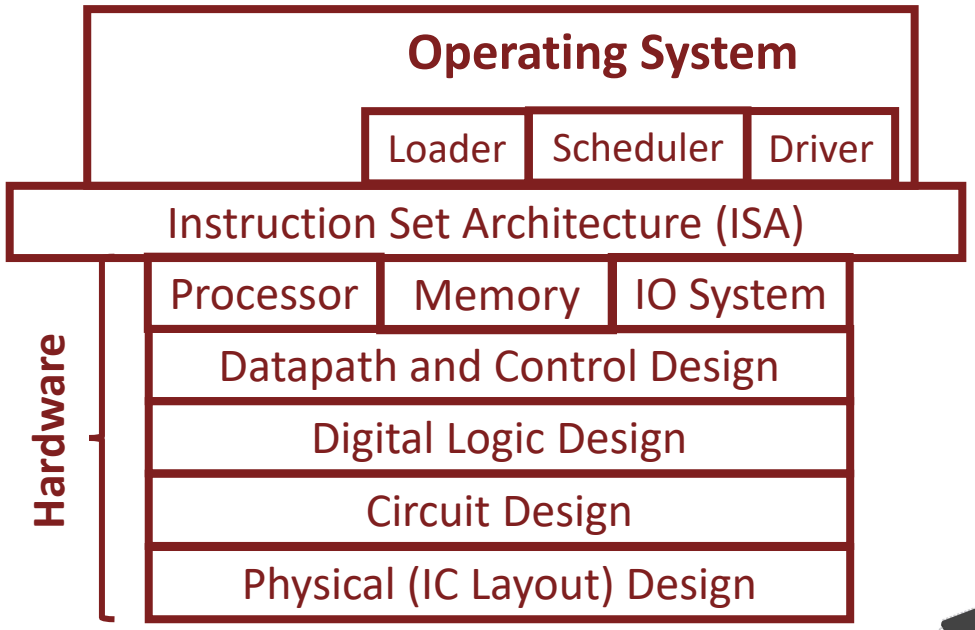


Instruction	Instruction Code	Operation
NOP	0000 0000	No operation
LDAC	0000 0001 Γ	*AC*←M[Γ]
STAC	0000 0010 Γ	M[Γ]←*AC*
MVAC	0000 0011	*R*←*AC*
MOVR	0000 0100	*AC*←*R*
JUMP	0000 0101 Γ	GOTO Γ



Computer System – Full System Stack

- General Model of a Computer System

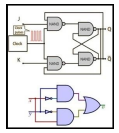


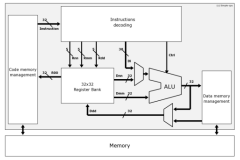
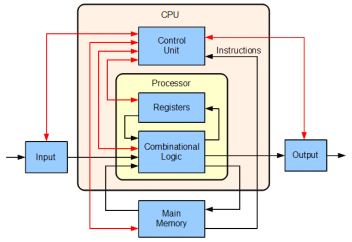
```
int square(int num) {
    return num * num;
}
```

```
square:
        push    rbp
        mov     rbp, rsp
        mov     DWORD PTR [rbp-4], edi
        mov     eax, DWORD PTR [rbp-4]
        imul    eax, eax
        pop     rbp
        ret
```

Instruction	Instruction Code	Operation
NOP	0000 0000	No operation
LDAC	0000 0001 Γ	*AC*←M[Γ]
STAC	0000 0010 Γ	M[Γ]←*AC*
MVAC	0000 0011	*R*←*AC*
MOVR	0000 0100	*AC*←*R*
JUMP	0000 0101 Γ	GOTO Γ

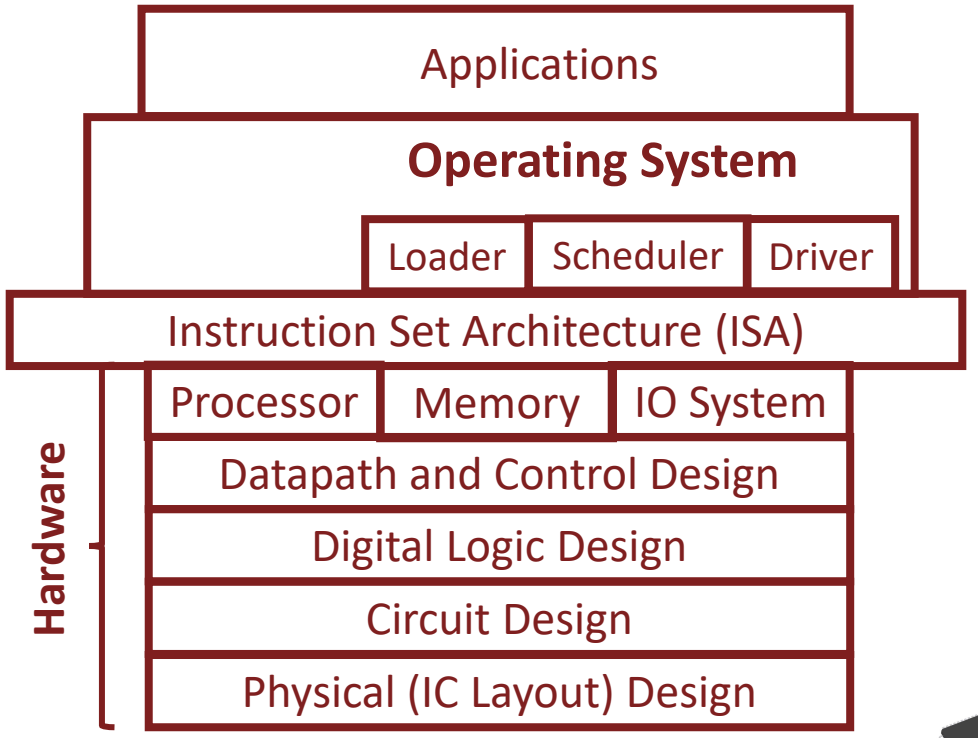




Computer System – Full System Stack

- General Model of a Computer System

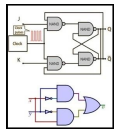


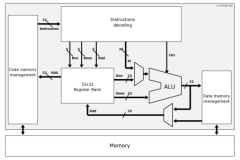
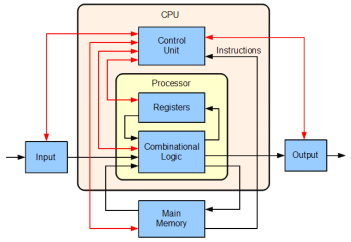
Instruction	Instruction Code	Operation
NOP	0000 0000	No operation
LDAC	0000 0001 Γ	*AC*←M[Γ]
STAC	0000 0010 Γ	M[Γ]←*AC*
MVAC	0000 0011	*R*←*AC*
MOVR	0000 0100	*AC*←*R*
JUMP	0000 0101 Γ	GOTO Γ

```
2   int square(int num) {
3       return num * num;
4   }
```

```
1   square:
2       push    rbp
3       mov     rbp, rsp
4       mov     DWORD PTR [rbp-4], edi
5       mov     eax, DWORD PTR [rbp-4]
6       imul    eax, eax
7       pop     rbp
8       ret
```

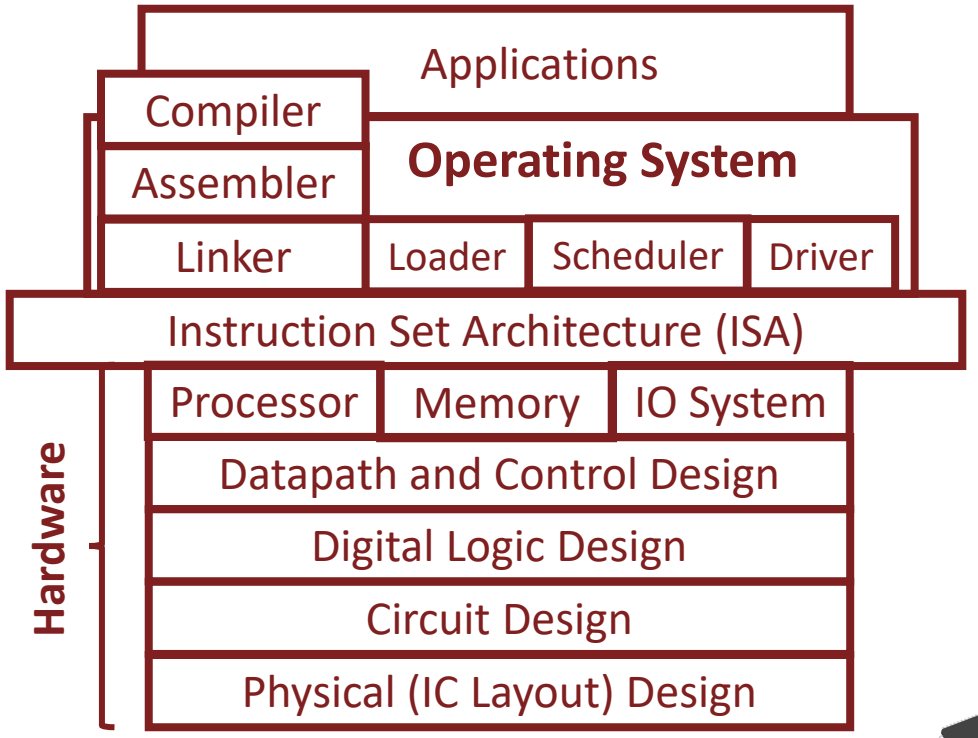




Computer System – Full System Stack

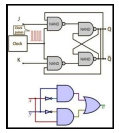
- General Model of a Computer System

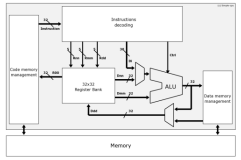


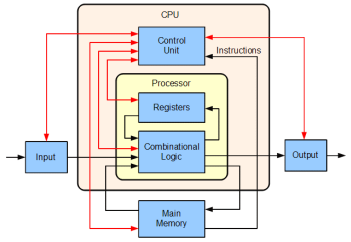
Instruction	Instruction Code	Operation
NOP	0000 0000	No operation
LDAC	0000 0001 Γ	*AC*←M[Γ]
STAC	0000 0010 Γ	M[Γ]←*AC*
MVAC	0000 0011	*R*←*AC*
MOVR	0000 0100	*AC*←*R*
JUMP	0000 0101 Γ	GOTO Γ

```
2  int square(int num) {
3      return num * num;
4  }
```

```
1  square:
2          push    rbp
3          mov     rbp, rsp
4          mov     DWORD PTR [rbp-4], edi
5          mov     eax, DWORD PTR [rbp-4]
6          imul    eax, eax
7          pop     rbp
8          ret
```

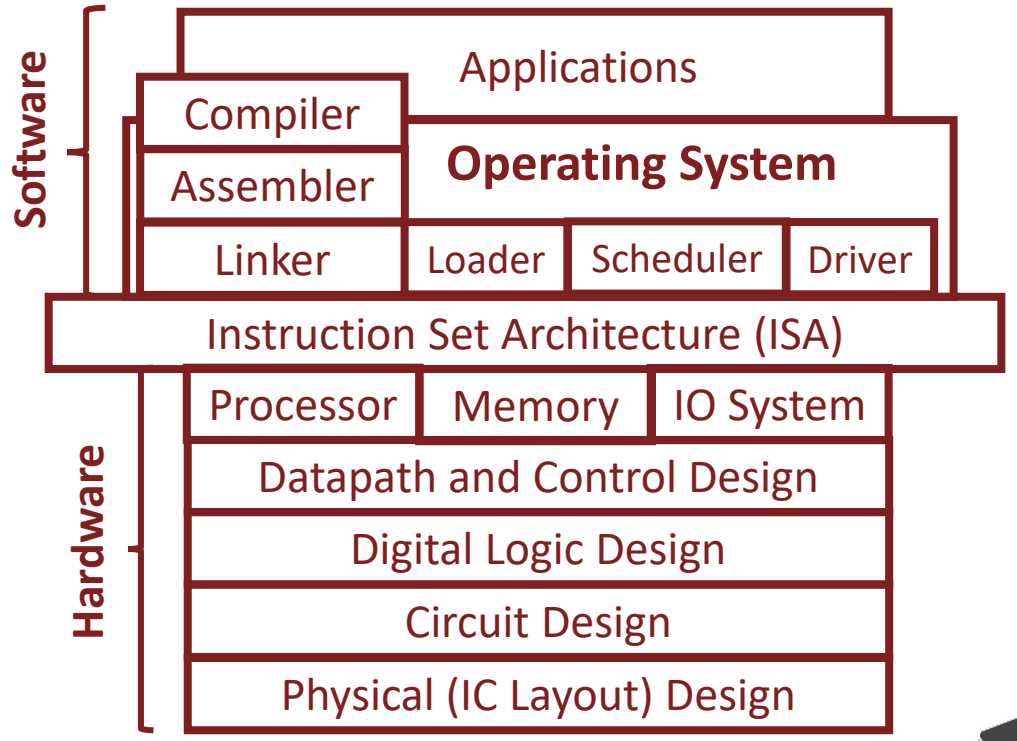


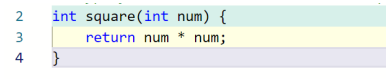




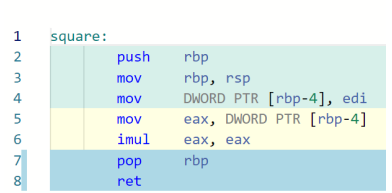
Computer System – Full System Stack

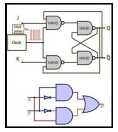
- General Model of a Computer System

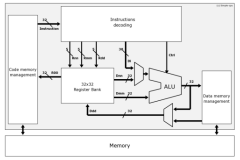


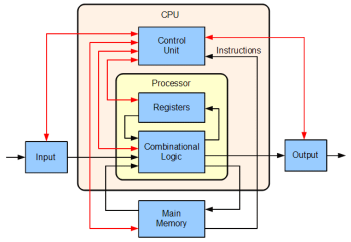


Instruction	Instruction Code	Operation
NOP	0000 0000	No operation
LDAC	0000 0001 Γ	*AC*←M[Γ]
STAC	0000 0010 Γ	M[Γ]←*AC*
MVAC	0000 0011	*R*←*AC*
MOVR	0000 0100	*AC*←*R*
JUMP	0000 0101 Γ	GOTO Γ

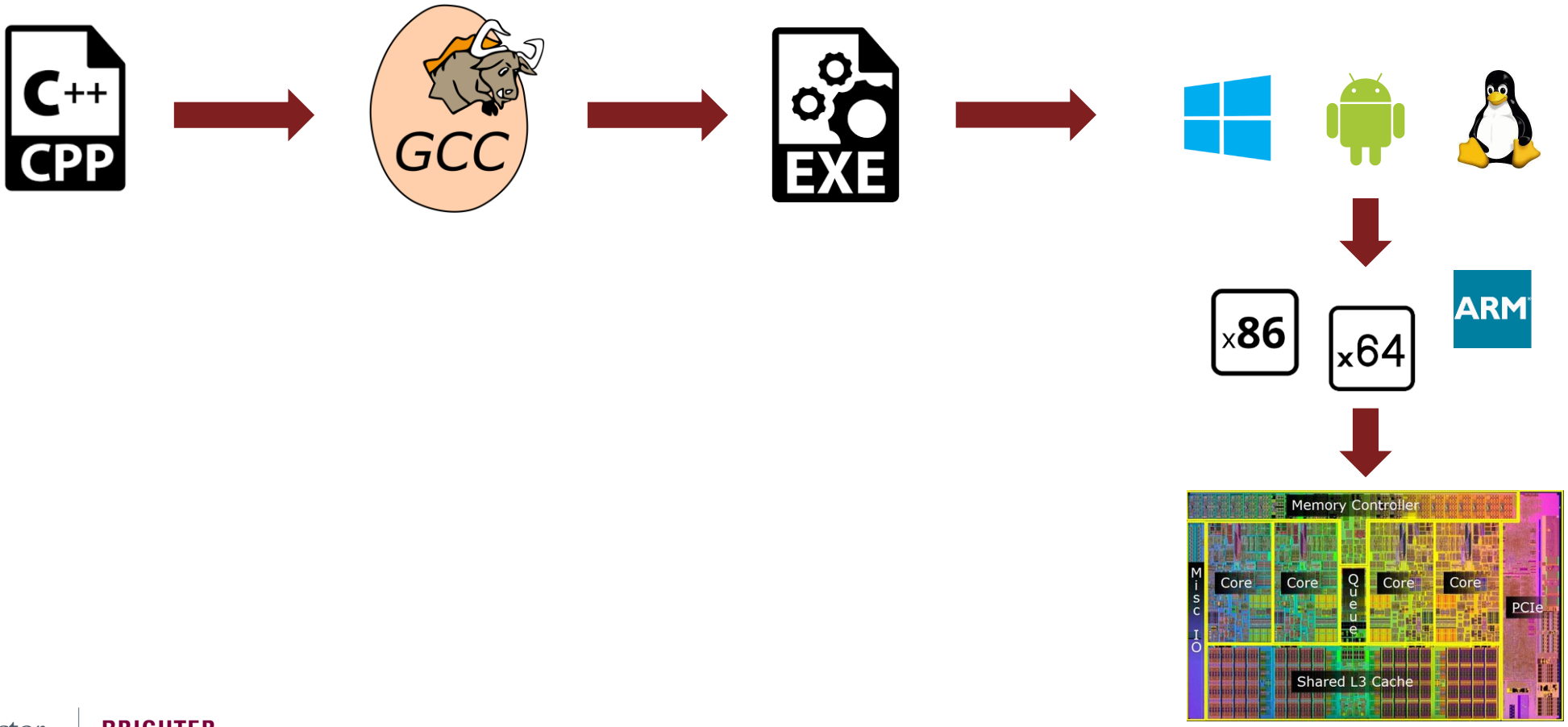




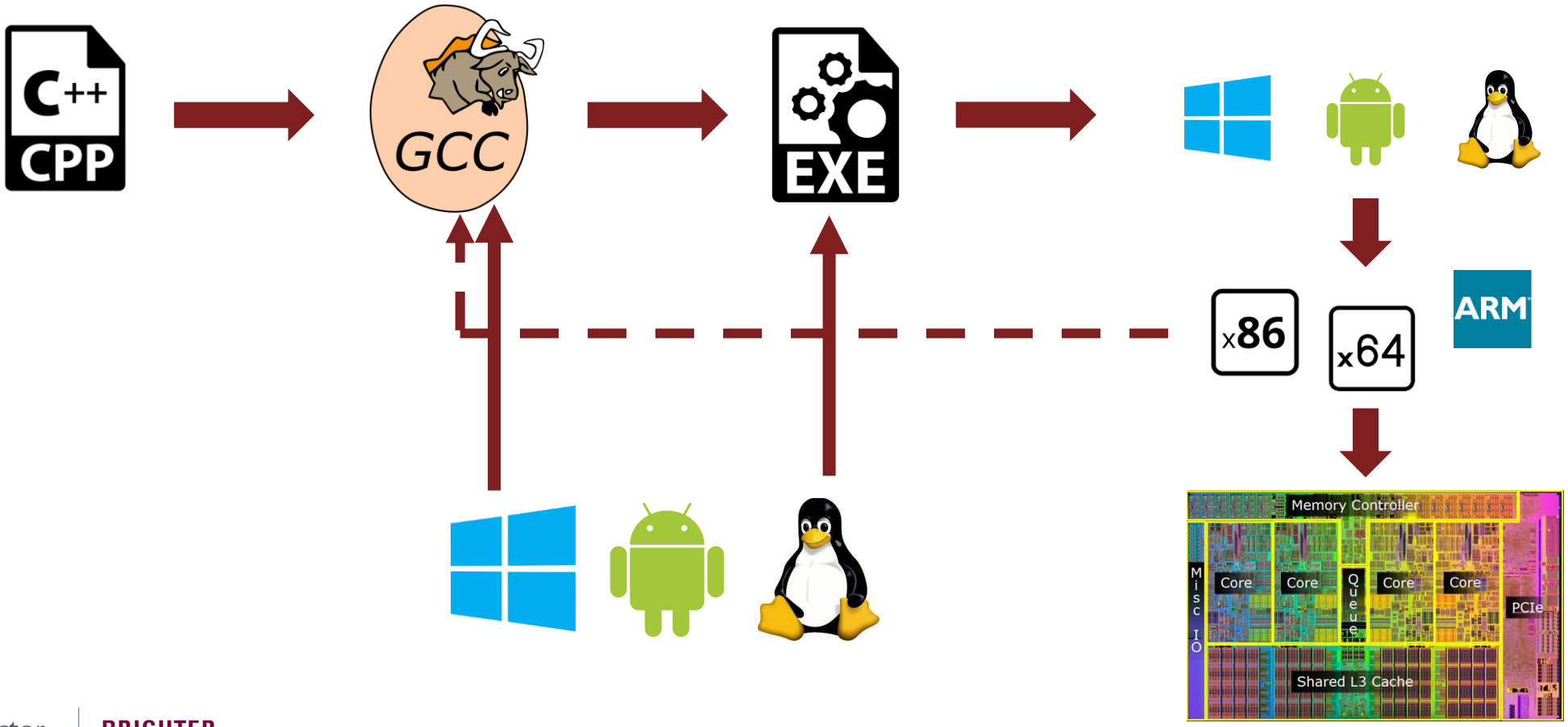




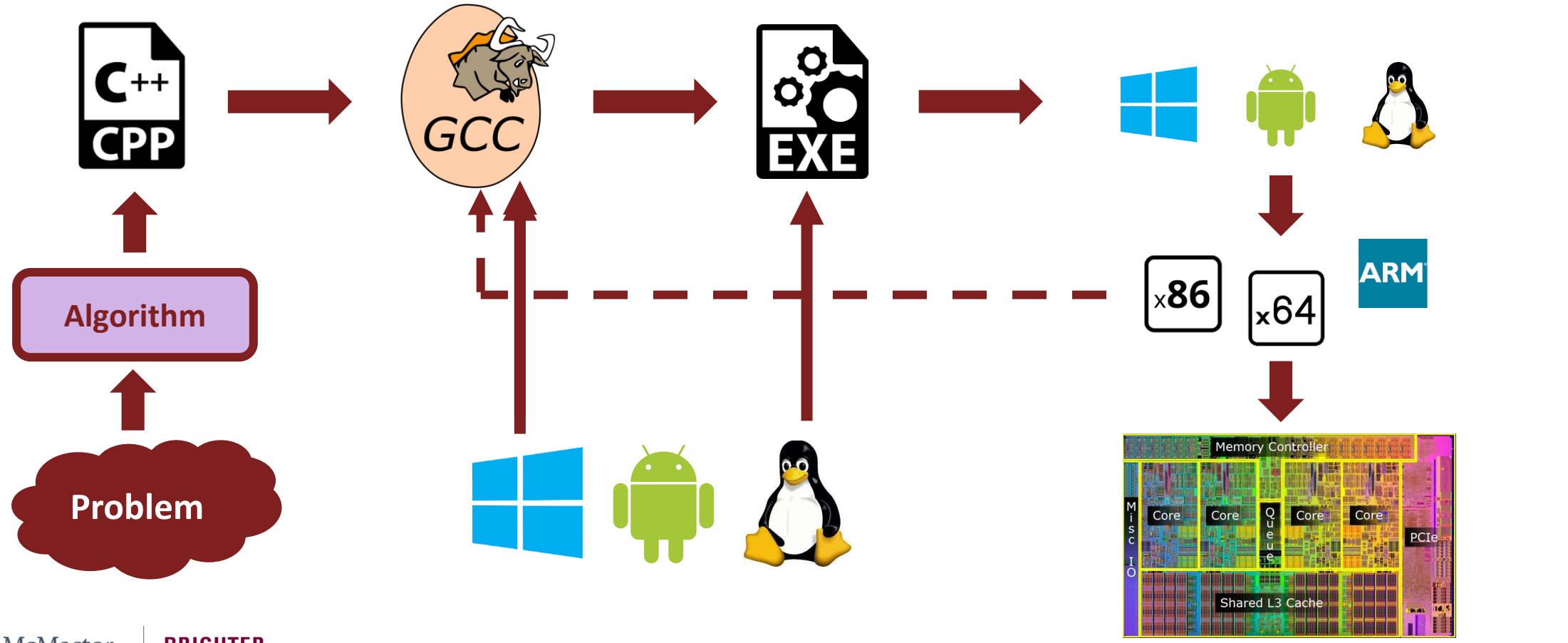
Computer System – Journey of a Program



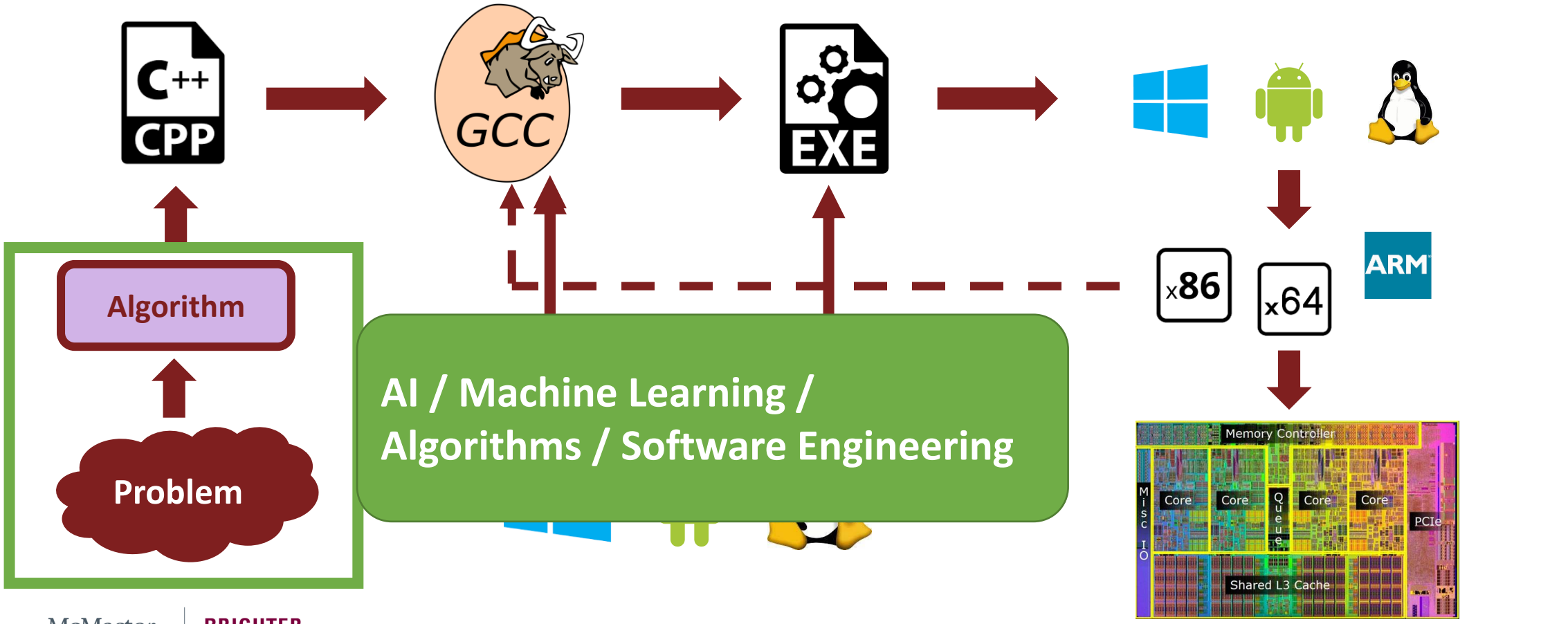
Computer System – Journey of a Program



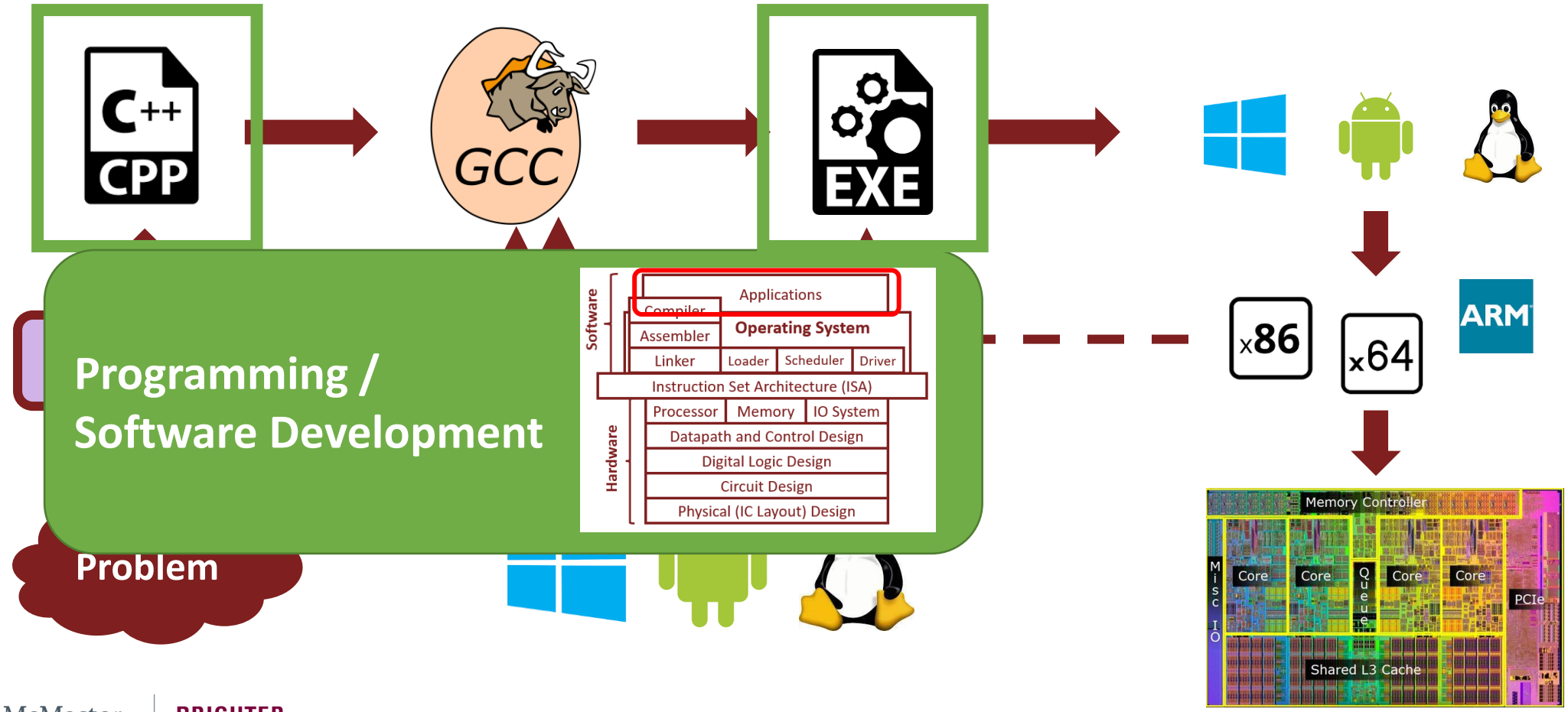
Computer System – Journey of a Program



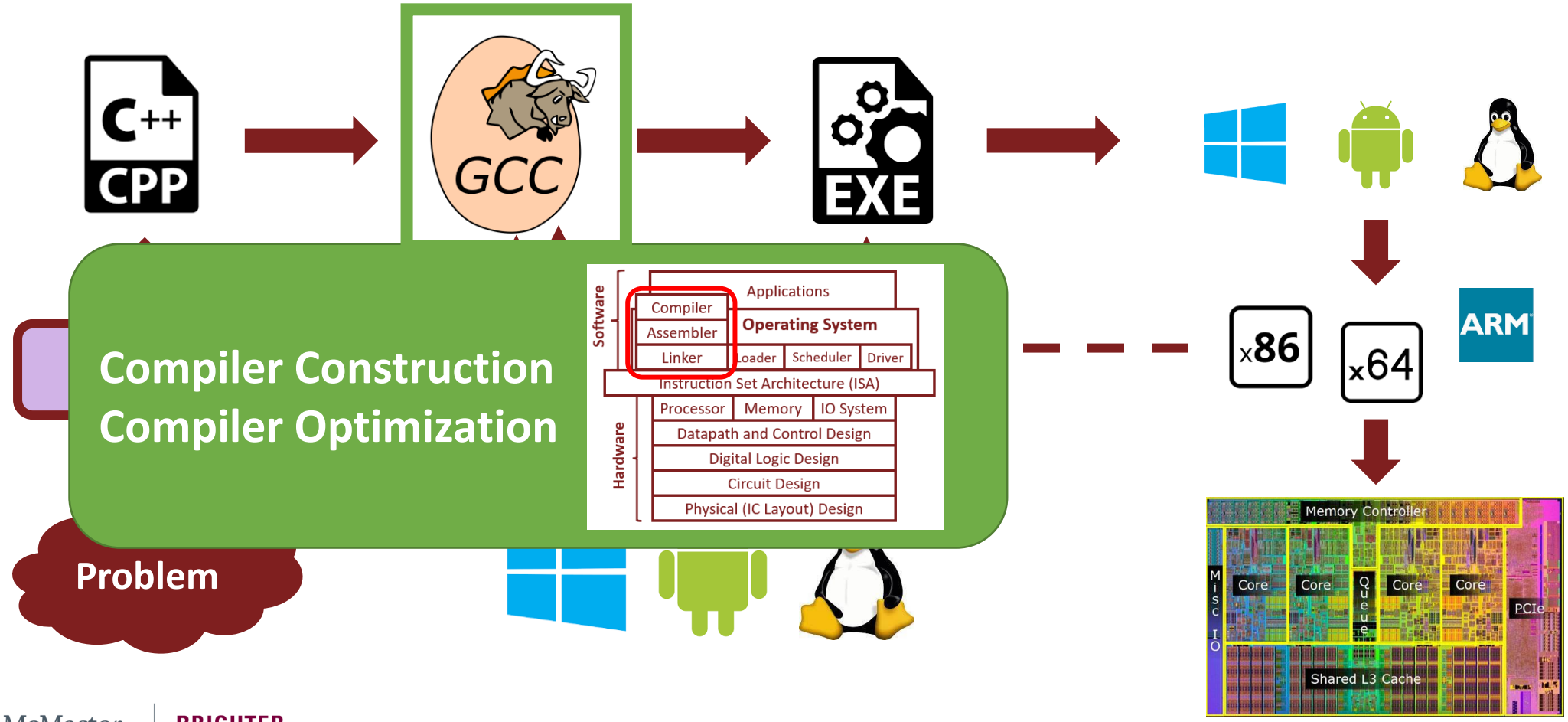
Computer System – Software Layer



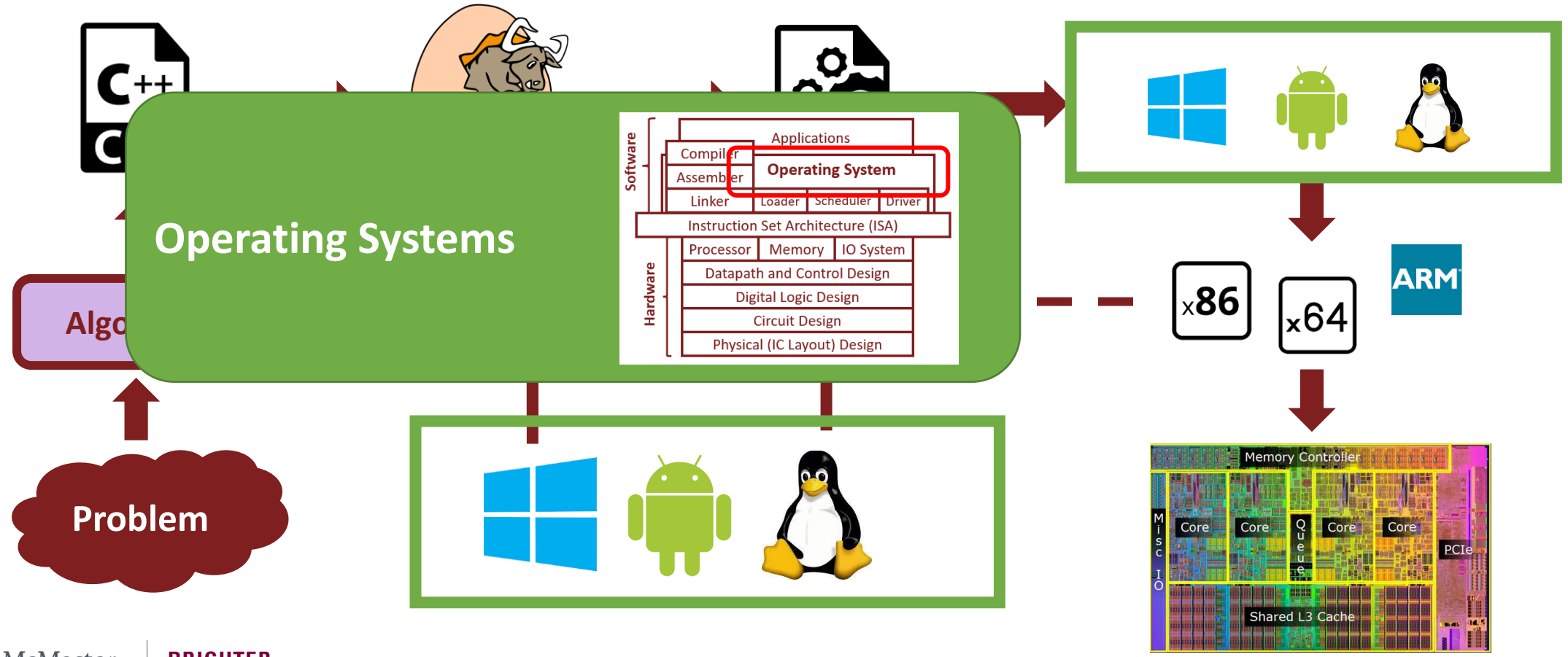
Computer System – Software Layer



Computer System – Software Layer

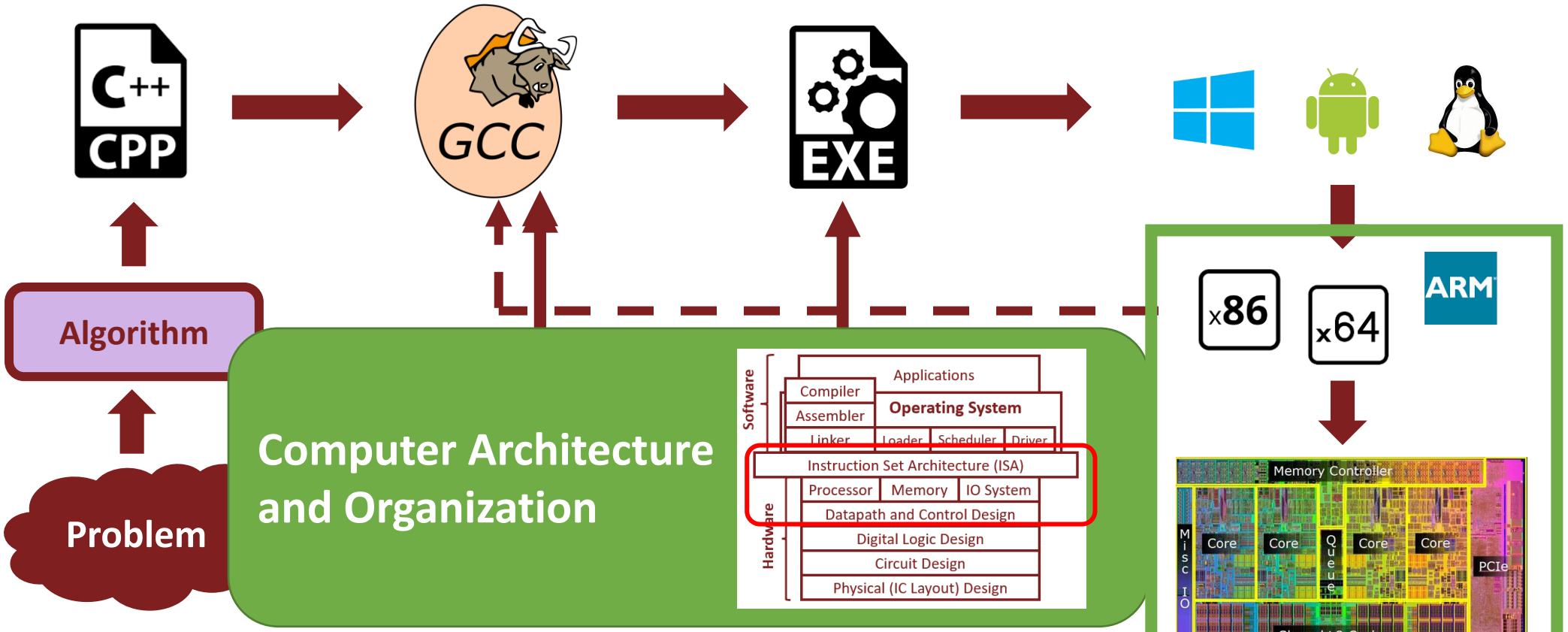


Computer System – Software Layer

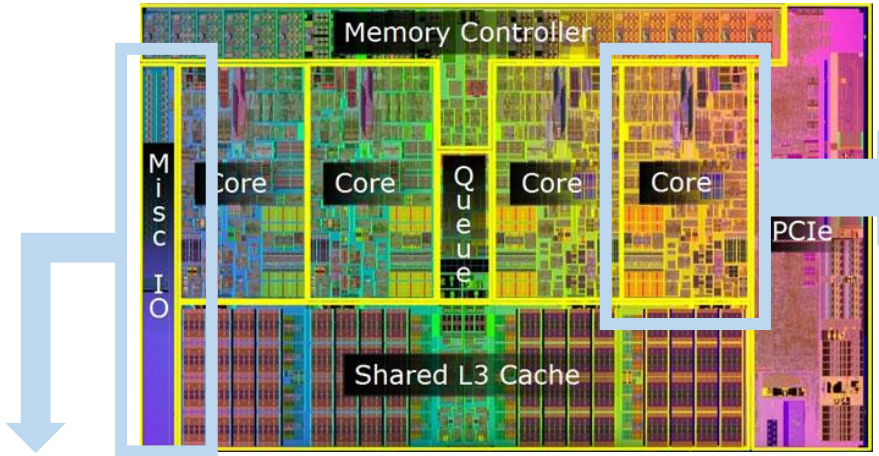


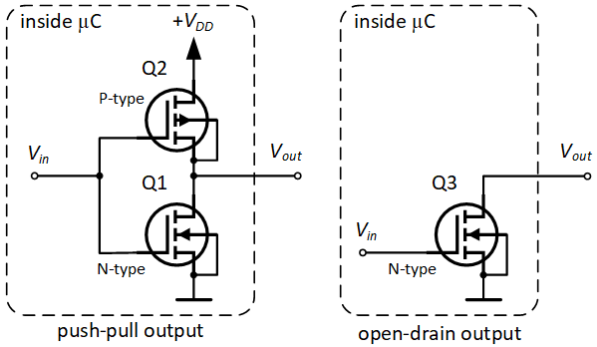
McMaster University | BRIGHTER WORLD

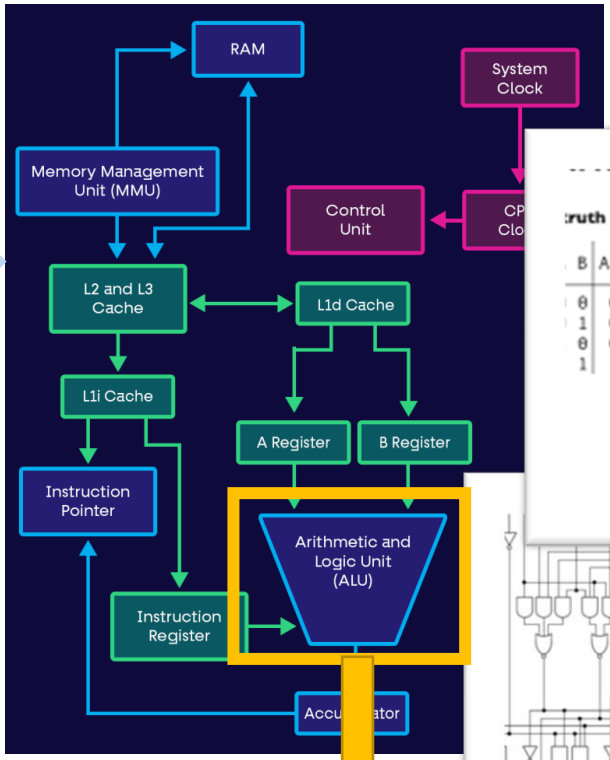
Computer System – Hardware Layer

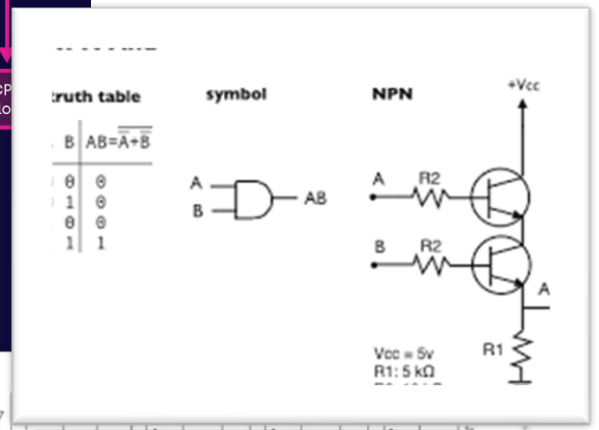


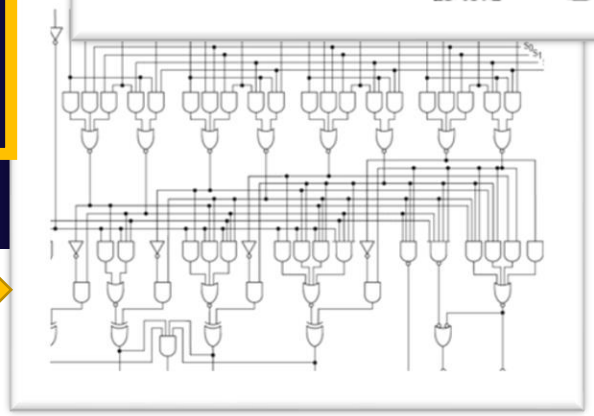
Computer System – Hardware Layer





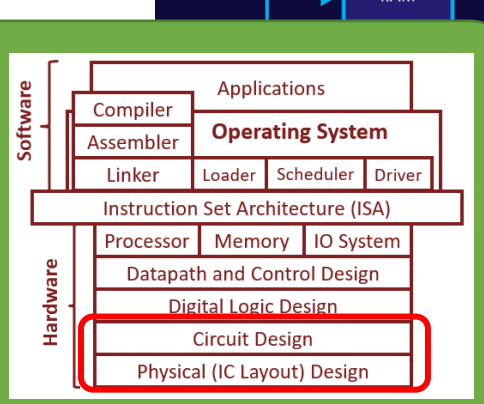


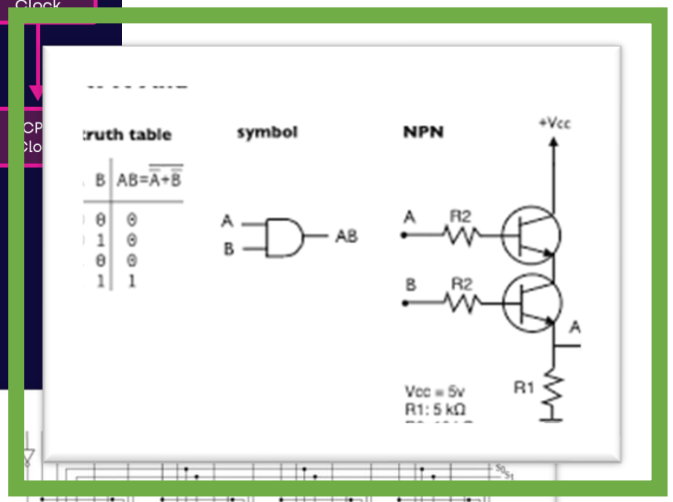


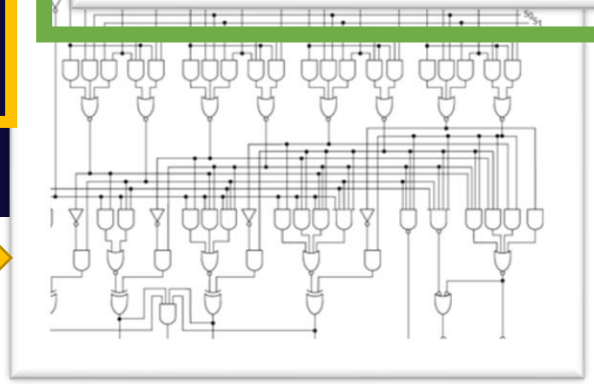


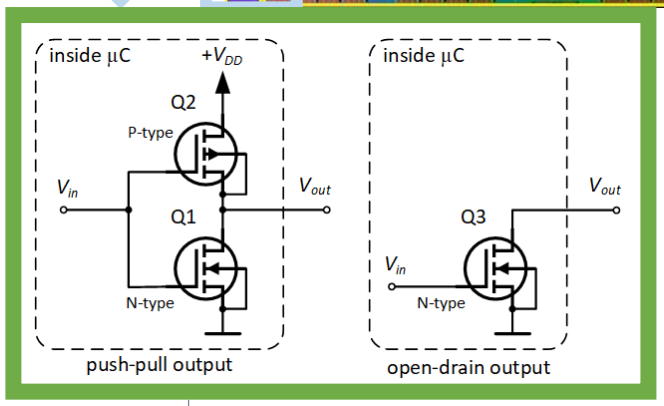
Computer System – Hardware Layer

Circuit Design
Electronics
Physical Design



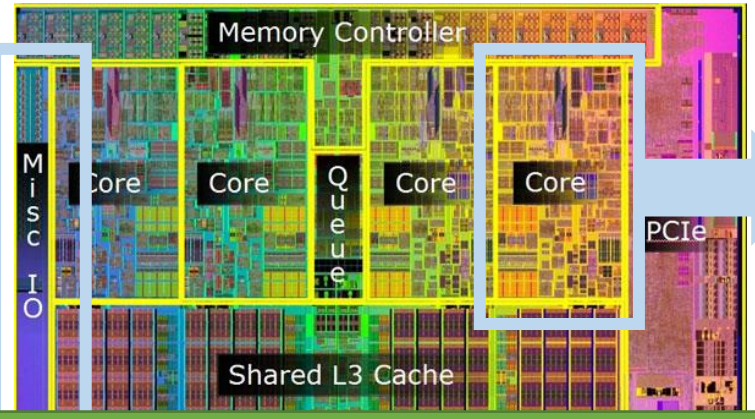


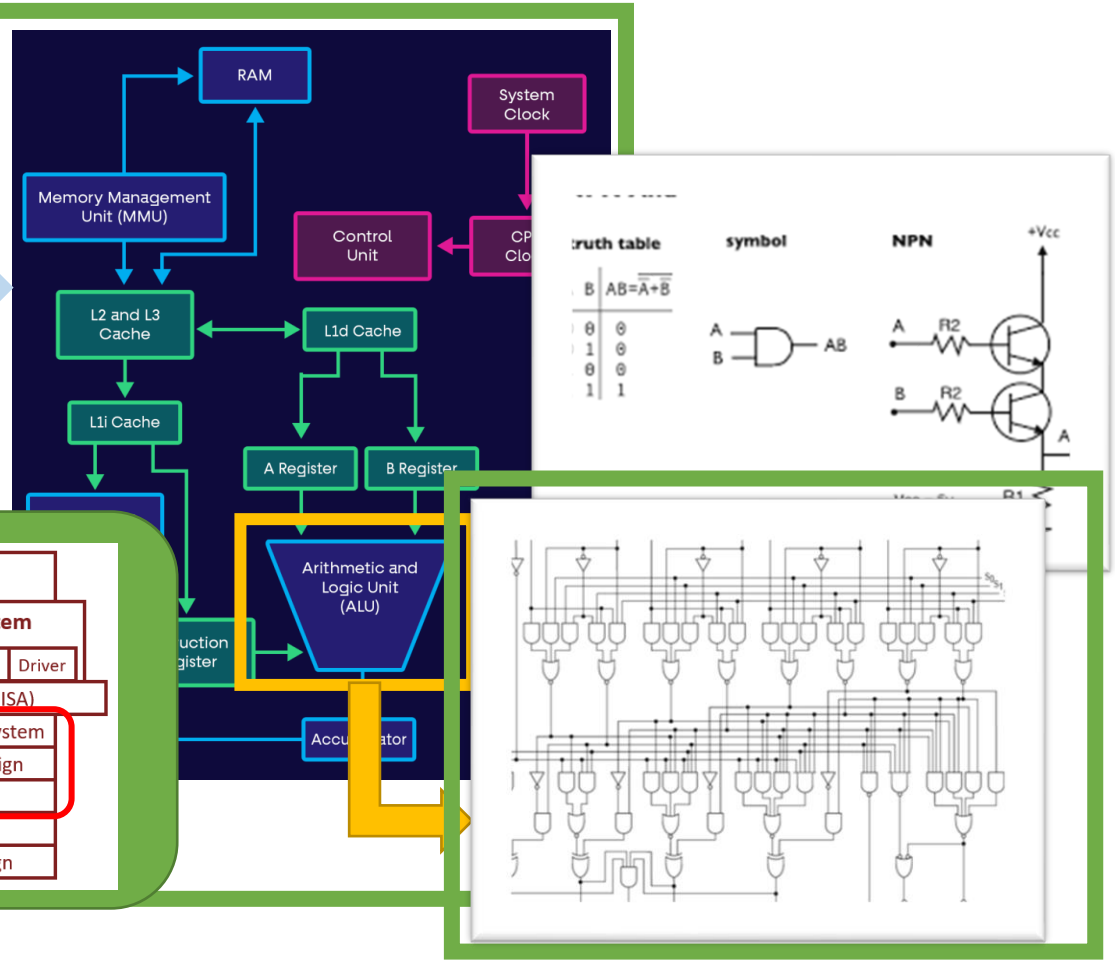




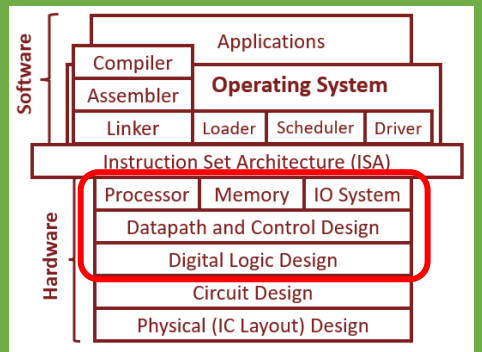
McMaster University | BRIGHTER WORLD

Computer System – Hardware Layer





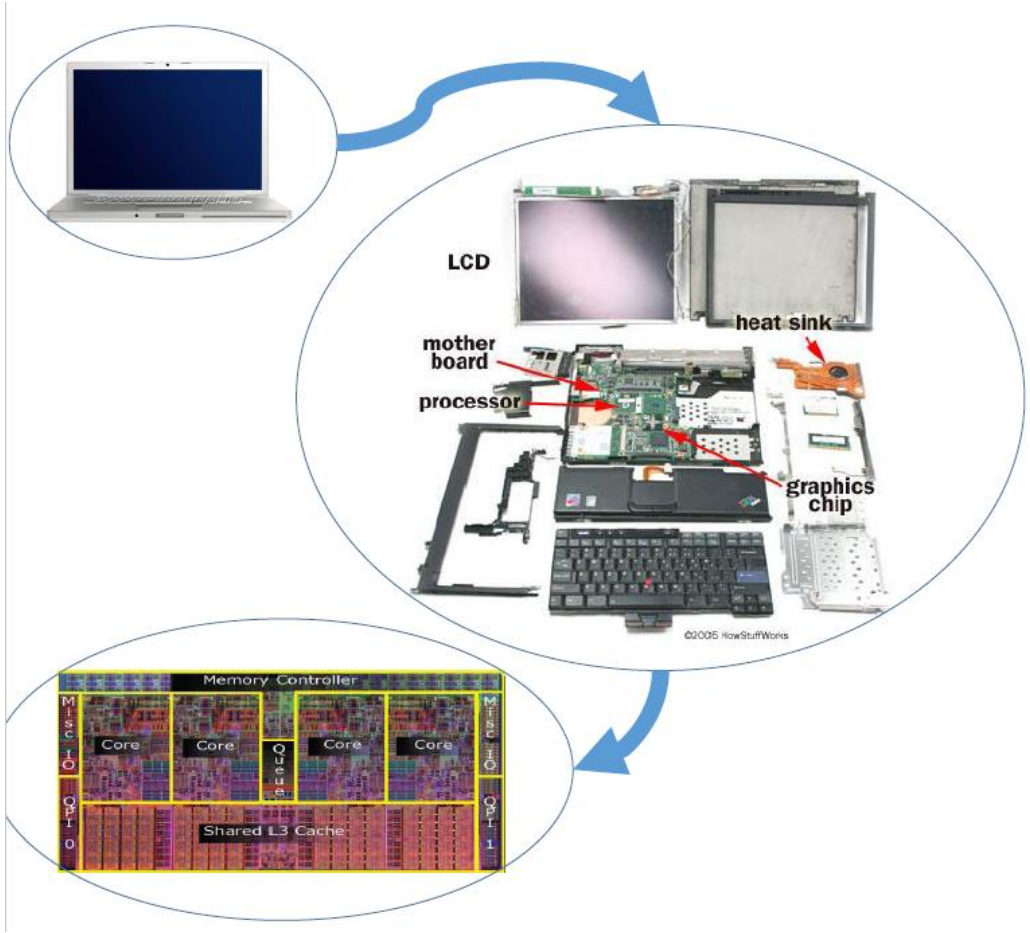
Digital Logic Design
CPU Architecture

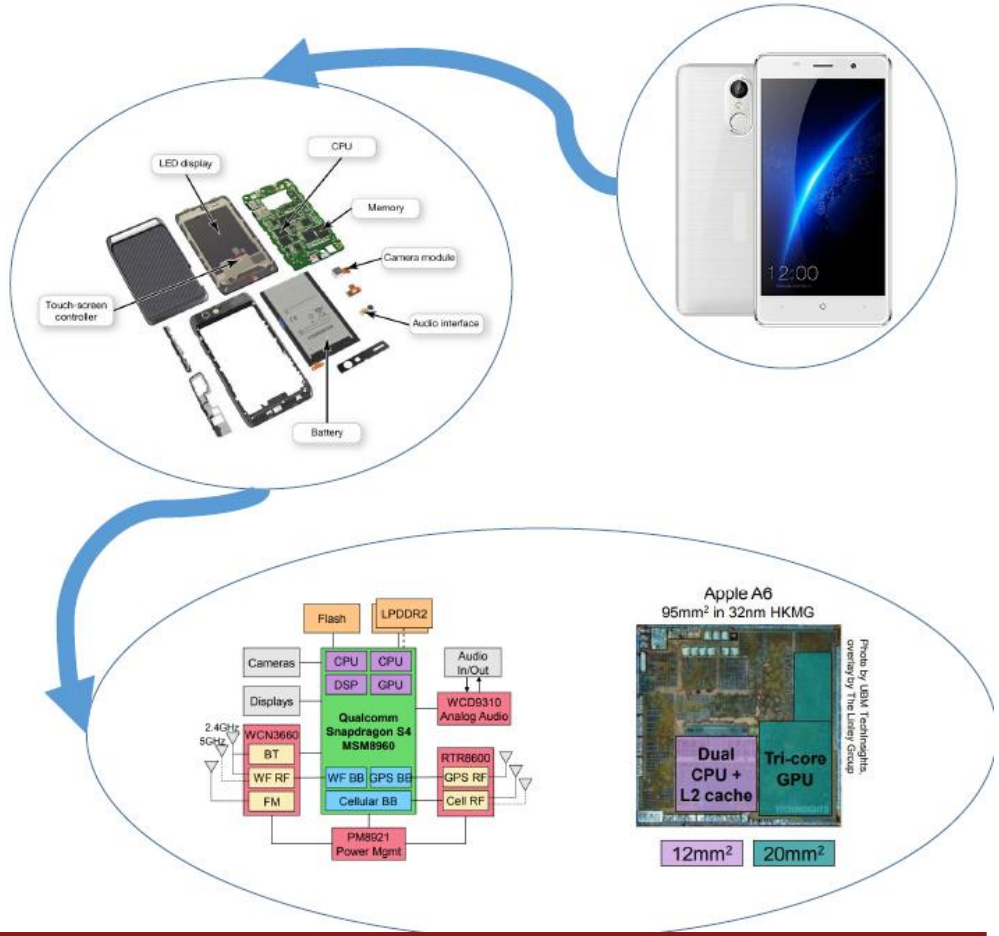


push-pull output open-drain output

McMaster University | BRIGHTER WORLD

Same Concepts, Different Shapes





Housekeeping

McMaster University | BRIGHTER WORLD

Our Teaching Team

- **Dr. S. Athar (Sec. C01)**
- **Dr. S. Chen (Sec. C02)**

- Teaching Assistants

Amrutha Haridas	harida1@mcmaster.ca
Muhammad Ahmed Kamran	kamram6@mcmaster.ca
Xiaomeng Liu	liu430@mcmaster.ca
Mina Nematian Doost	nematiam@mcmaster.ca
Ali Yousefi Darani	yousea11@mcmaster.ca
Fengyang Sun	sunf10@mcmaster.ca
Christian Schaible	schaiblc@mcmaster.ca
Sai Vasavi	seelamvs@mcmaster.ca
Azadeh Gholaminejad	gholaa3@mcmaster.ca
David Boah Gyasi	gyasid1@mcmaster.ca
Ruchin Liang	liangr19@mcmaster.ca

Course Schedule (Sec. C01)

- **Lectures 5:30pm – 6:20pm**
  - **Mondays PGCLL 127**
  - **Wednesdays PGCLL M21**
  - **Fridays ABB 102**
  - All lectures are in-person unless otherwise announced on Avenue. Attend Lectures!

- **Tutorials – Thursdays 1:30pm – 2:20pm BSB B136**
  - Lab briefing and tips
  - Practice problems and programming tips

- **Labs/Project - Check Your Assigned Days! 2:30pm – 5:20pm**
  - **Tuesdays ETB 224/227/228 + JHE329 Thursdays JHE329**
  - Total of 4 labs + lab 0 for environment setup
  - GitHub classroom for lab distribution and submission
  - Read Lab 0 manual for all submission details

New Version of COMPENG 2SH4

- Experiential Learning with semi-Flipped Classroom
  - Every lecture is accompanied by one or two 10 to 15-minute pre-class video(s)
  - Live coding activities in almost every class
  - **4 Labs** are designed to deliver Test-Driven Software Development experience
  - **1 Project + 3 Project Preparation Activities** are designed to deliver Iterative Software Design experience

New Version of COMPENG 2SH4

- Course Learning Goals
  - Developing fundamental understanding and skills in procedural and object-oriented programming
  - Developing good software design and debugging skills
  - Drive software designs from scratch using effective design tools

  - COE2SH4 is not going to be an easy course – but it will make your future courses much easier.
    - Data structures and algorithms, Operating Systems, Software Design Patterns, etc.
    - All subsequent (D-designated) design courses

- Career Preparation Goals
  - Good iterative engineering design practice
  - Good problem analysis and solution validation practice
  - Speaks the same industry language

Labs

- Labs are your first-hand experience in Test-Driven Software Development
  - Each Lab is designed with est. workload of 2-4 hours.
- Everyone has a pre-designated lab section.
  - You are only allowed to attend your designated lab session due to space limitation
  - **HOWEVER**, you can attend any of the TA office hours for help
- Each lab runs for exactly **2 weeks**, and **due on Friday of the second week**
  - Always check deadlines on GitHub classroom
  - Usually, tutorials of the first lab week will contain Lab Briefing and Tips
- Strategy for Success
  - Start the lab right away – do not wait until the last minute
  - Briefing Tutorial is very helpful. Attend it.
  - Use TA office hours and your designated lab session to solve challenges
    - Do **NOT** count on these sessions just to get your lab started

Project and Project Preparation Activities

- To gain experience in Iterative Software Design
  - **Three 2-Week Project Preparation Activities**
  - **One 4-Week Design Project**

- Project Preparation Activities (PPAs)
  - There are 3 PPAs running in parallel with Lab1 – Lab 3
    - Each PPA has an est. workload of 4-6 hours
  - All 3 PPAs contain *detailed design steps* and tips to help you get through the activities.
    - Following the instructions may greatly reduce your workload.
  - Each PPA will yield a piece of interactive software that incrementally gets better and better.
    - Eventually contributing to your design project.
  - Each PPA runs in sync with its labs for exactly **2 weeks**, and **due on Sunday of the second week**
    - You are required to give a **5-minute demo** at your designated lab session after the due date

Project and Project Preparation Activities

- To gain experience in Iterative Software Design
  - Three 2-Week Project Preparation Activities
  - One 4-Week Design Project
- Course Project – Group of 2
  - 4-week project in November
  - Work done in PPAs completes 60% of the implementation
  - Project Evaluation
    - Functional Evaluation – by Teaching Team
    - Peer Evaluation and Project Reflection – by Student Groups
- Strategy for Success
  - Read the manuals, watch the training videos, attend live coding activities in class
  - Respect Incremental Engineering
  - Use TA office hours and your designated lab session to solve challenges

Flipped Classroom

- Pre-Lecture Videos (Average 10-15 min long)
  - Theory-focused, animated
  - **All lectures are executed assuming you've watched the pre-lecture videos**
- Activities in Class (and Tutorials)
  - Theory deep-dives, interactive Q&As
  - Simple coding activities for theory verification
  - Extended coding activities for programming and software design skills development
  - PPA coding guidance + Guided software sprints for Project
  - Attending lectures and tutorials is highly recommended.
- Post-Lecture Weekly Problem Sets
  - Ungraded, application-focused, great for concept validation and exam preparation

Marking Scheme

- Labs 18 %
  - 2% for Lab 0
  - 16% for Lab 1 – 4, 4% each
- Project 22 %
  - 4% for each Project Preparation Activity (PPA)
  - 10% for Project
- Midterm 25 %
- Final 35 %
- Refer to the course outline for more details

We are here to help!

- **TA Office Hours (Week 2 onward)**
  - 2-hour office hours on daily basis @ **ITB AB105**
  - **10:30am – 12:30pm**

- **Instructor Office Hours (Week 2 onward)**
  - Mondays and Wednesdays 2:00pm – 4:00pm @ ITB A317 (in-person)
  - Additional office hours may be arranged upon email request only if TAs are unavailable

- **Communication Protocol**
  - Always email with ***[COMPENG 2SH4]*** prefix in your subject
    - Otherwise, your email will be filtered out by my outlook subject filter.
  - I will strive for 48-hour turnaround time

Tentative Instructional Plan

Week	Date	Key Topics		Labs	Project
1	Sep 3 – Sep 6	Introduction to C, Simple C Program Examples, Data Types		Lab 0	
2	Sep 9 – Sep 13	C Functions, Prototypes, Scope of Fields		Lab 0	
3	Sep 16 – Sep 20	Primitive Data Types and Operators, Control Structures, Console IO		Lab 1	PPA1
4	Sep 23 – Sep 27	Arrays, Strings, Qualifiers		Lab 1	PPA1
5	Sep 30 – Oct 4	Debugging Principles, Structs, Unions, Enumerations, and Functional Organizations		Lab 2	PPA2
6	Oct 7 – Oct 11	Pointers, Pass by Reference / Value, Pointers and Primitive Arrays		Lab 2	PPA2
Recess	**Oct 14 – Oct 18**	**Midterm Recess, No Classes**			
7	Oct 21 – Oct 25	Dynamic Memory Allocation, Array of Pointers	**MIDTERM**	Lab 3	PPA3
8	Oct 28 – Nov 1	C Modularizations, Memory Management Tips, Introduction to C++		Lab 3	PPA3
9	Nov 4 – Nov 8	OOD, C++ Classes Implementation, Rule-of-Six, UML Class Representation, **Formal Project Briefing**		Lab 3	PPA3
10	Nov 11 – Nov 15	Array list, Array list applications		Lab 4	Project
11	Nov 18 – Nov 22	Inheritance		Lab 4	Project
12	Nov 25 – Nov 29	Recursion, Project Sprint			**Project**
13	Dec 2 – Dec 5	Polymorphism. Introduction to C++ Template Classes			**Project**

Rules and Regulations

- AI tools are prohibited.
- Zero tolerance for any form of plagiarism and academic misconduct
- All your work will be checked against
  - Your peers' work
  - Works from prior years
- Please refer to the regulations outlined in the student handbook and course outline regarding academic misconduct

Self-Learning Resource

- GNU C/C++ Reference Manuals
- ASCII Table
- Many Online Programming Communities
- Refer to Avenue course shell for additional recommendations

Tips for Success

- Attend class lectures
- Attend tutorials
- Work on labs and project, and invest hours into them
- When in doubt, ask for help!
- Do not leave things till the last minute
  - Programming is **Incremental Engineering Design**, not essay writing

Upcoming

- **Simple C Programs**
  - And how it differs from Python
- **C Compilation Process**