Course Organization

- Top-down approach to microprocessor programming and design.
- Lectures focus on structured computer organization, and progress through "layers" :
  - Instruction set architecture
  - Assembly language level + problem-oriented language level (embedded C)
  - Microarchitecture
  - Operating system level
- Application of design principles on state-of-art architecture
  - ARM Cortex M processor family
- The course focuses primarily on experimental work.
  - Usual lecture: 45 minutes on basics/theory and 30 minutes on how it applies to your hardware device and experiments.

Course Basics

- **Instructor:** Prof. Zeljko Zilic
  Room 546, McConnell
  Tel: 398-1834, Fax: 398-4470
  e-mail: zeljko.zilic@mcgill.ca

- **Office Hours**: Wed: 11:30-12:30, by appointment (after lectures) – can adjust later. If by e-mail, use **"ECSE 444"** in subject
- **Teaching Assistants:** Hamza Javed and Shahab Mahmoudi Sadaghiani

Course: Lab Structure

- 4 experiments + final project.
  - Experiment 1 : Assembly and C
  - Experiment 2: Intro. to hardware interfacing; drivers, timing …
  - Experiment 3-4: I/O, Interrupts, DMA, Advanced Sensor Use, Networking, …
- Project (example): Sensors, display + wireless interface between boards, novel applications, cloud interface, smartphone access, …

Organization/Administrivia

- Labs conducted in pairs –choose in a week
  - Team of 2 students: one report, but work graded individually
- Final project in group of 4
- 15 minute quizzes will be conducted during four lectures or tutorials in the term
  - lecture material since the previous quiz + most recently completed experiments
  - In-person or online, depending on circumstances
- **Tutorials will be announced within two weeks**
- Penalties for Late Assignments: 5 % per day (Fri-Mon=1day)
- Missed demo - reschedule for 65 percent of grade

Course Timeline

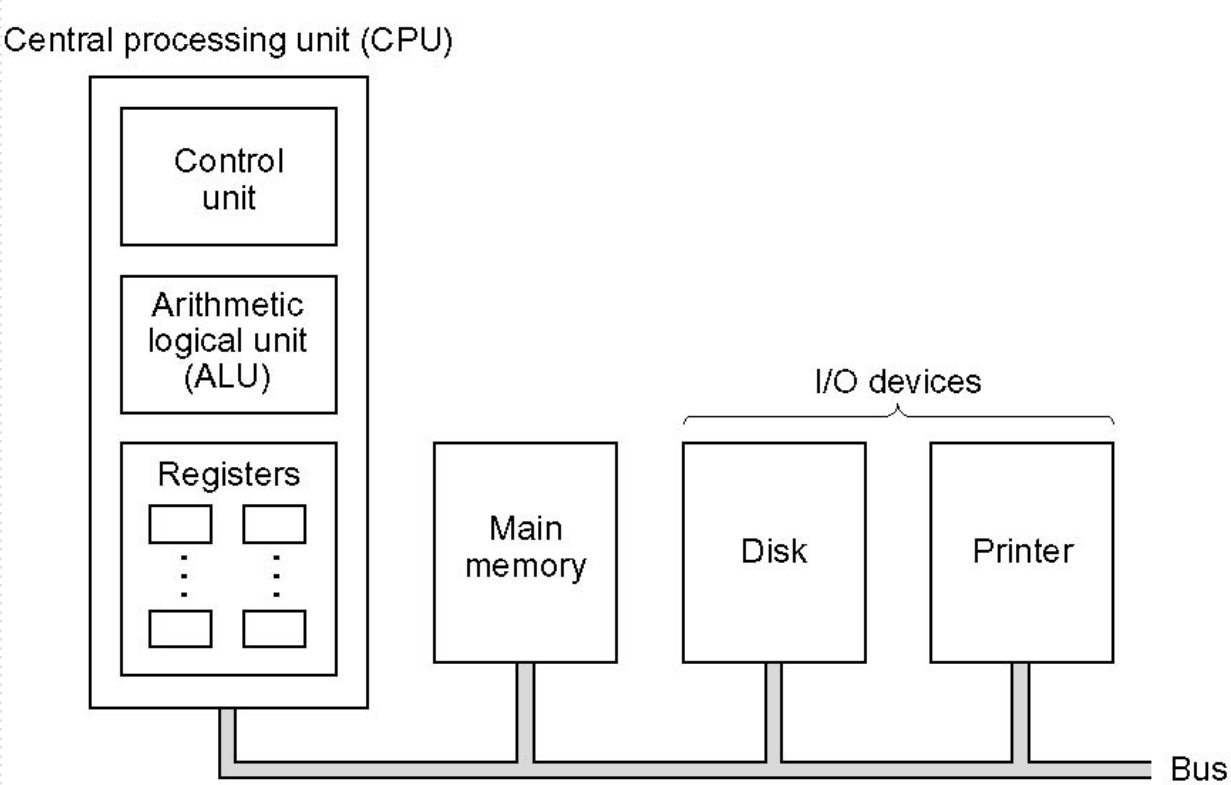
Week	Lecture	Tutorial	Experiment
1: Jan. 11	Intro, organization	Tools	Tutorial
2: Jan. 18	Instruction Set, Assembler	Assembly	1: Assembly +
3: Jan. 25	Instruction Set Arch.	Linking, Debug	1: Embedded C
4: Feb. 1	ISA ctd, Quiz 1	ADC,DAC,Timer	2: Peripherals
5: Feb. 8	I/O, Interfacing	Peripherals	3: Peripherals
6: Feb. 15	I/O, Interfacing, Quiz 2	Flash	4: I/O
7: Feb. 22	Microarchitecture	Interfacing	4: I/O
8: Mar. 8	Microarchitecture, Quiz 3		Project
9: Mar. 15	System Design Principles	Networking	Real time clock
10: Mar. 22	Design Principles, Quiz 4		LCD
11: Mar. 29	Sensing and Actuating		Project Progress
12: Apr. 5	Networking		Integration
13: Apr. 12	Security		Final Demo

Course Overview

- Background
  - Computer Arch. Basics
- Microprocessor Instruction Set Architecture
- Embedded Processors
- Embedded System Design
  - HW and SW techniques
- Building Real Systems
  - Techniques and Tools

Computer Organization

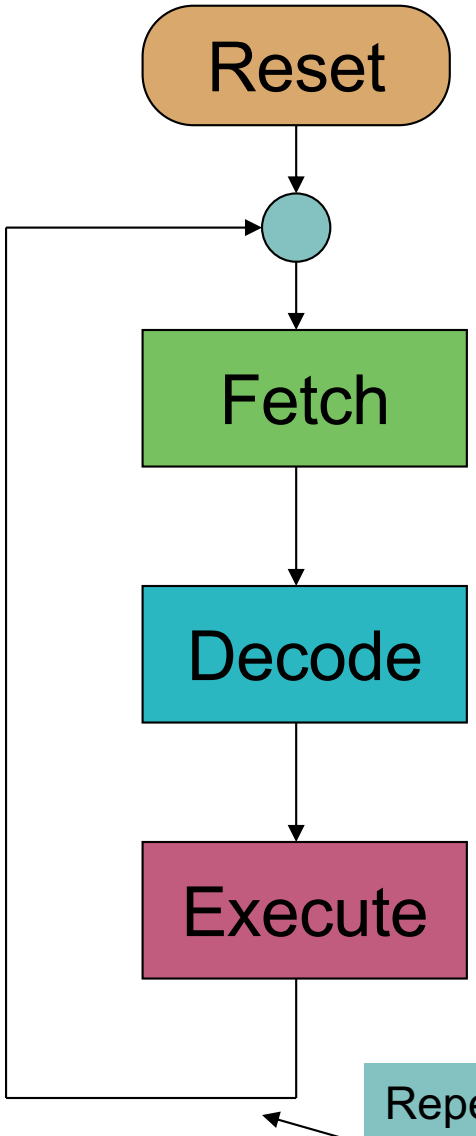
- Processor
  - Microprocessor
- Memory
- Peripherals
- Common Bus



Background Refresher

- Microprocessor basics
  - Operation
  - Some (far from all) optimizations
    - Aspects that you will face in course
- Software/system bringup
  - Interaction between HW and SW
    - To give a "feel" of things in the course
- Presentation style: "visual" plus "reference" slides, (plus blackboard?)

Microprocessor Operation



Start here at power-on or when a reset signal is received

1.Output inst. address on address bus

2. Read inst. pattern from memory onto data bus

3. Increment inst. pointer (program counter)

Determine what type of instruction was fetched

1. If necessary, read data from memory

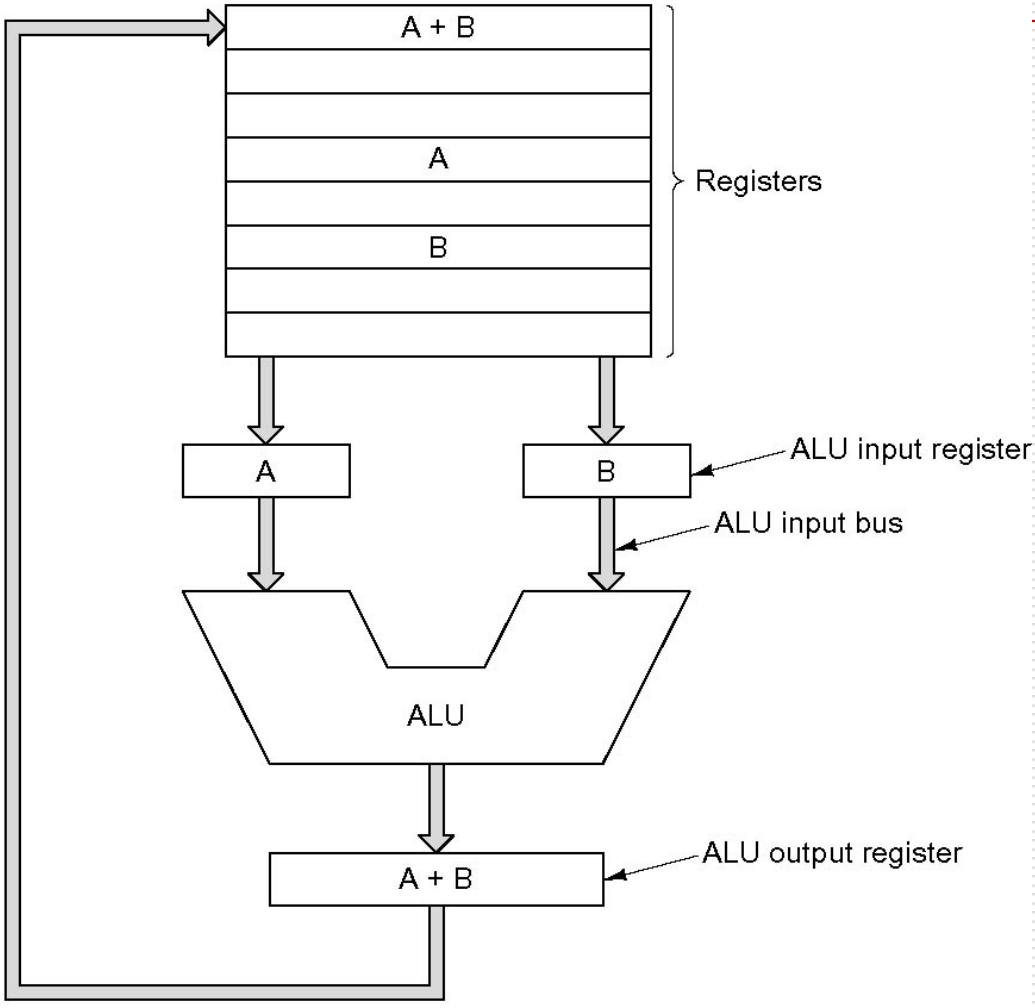
2. Execute instruction

3. if necessary, write results to memory

Repeat this process until power is turned off or the processor is halted.

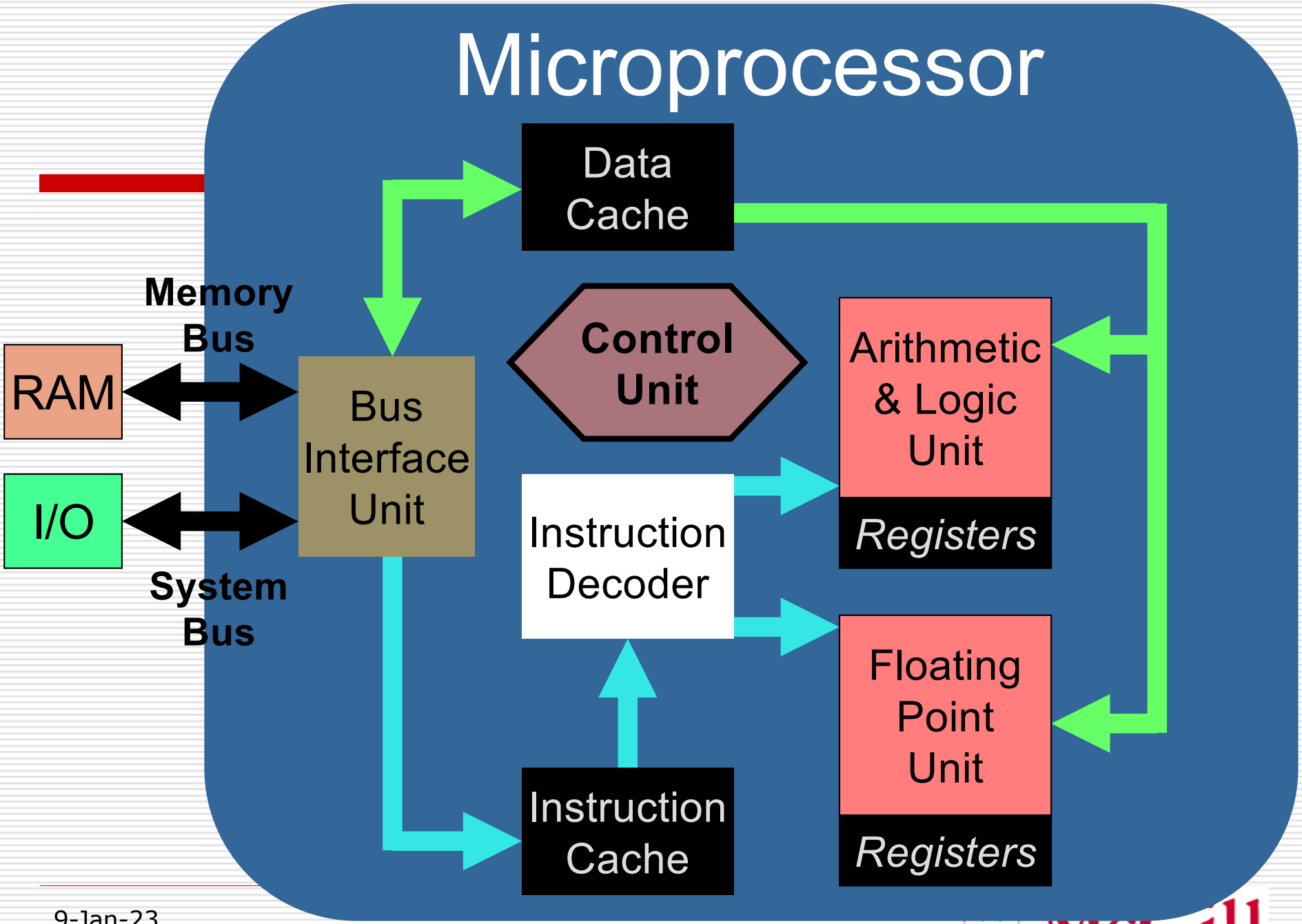
Common Processors

- Mainly von Neumann architecture
  - Arithmetic-logic unit
  - Registers
  - Auxiliary registers



Processor Execution - Java code

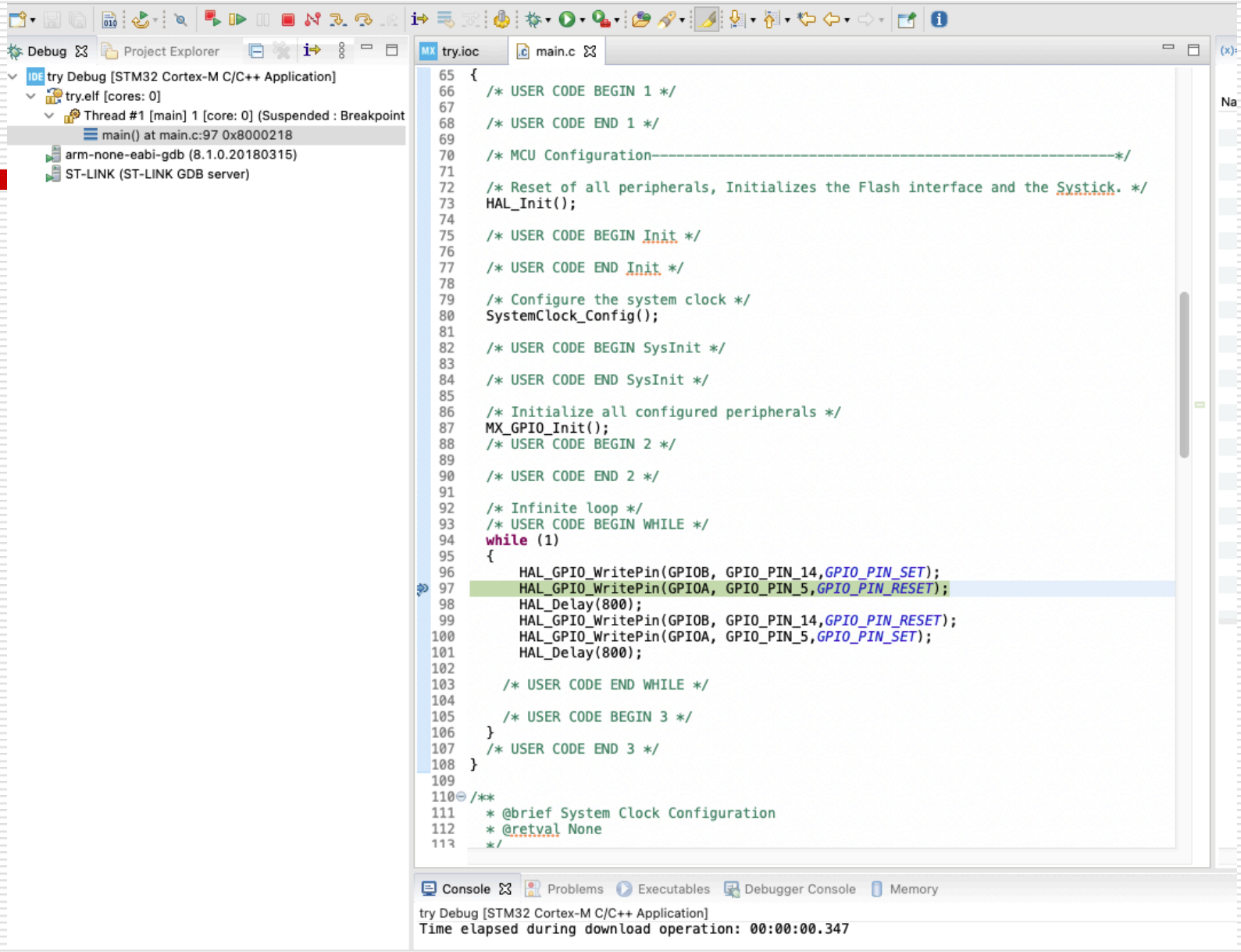
```
public class Interp {
static int PC;                          // program counter holds address of next instr
static int AC;                          // the accumulator, a register for doing arithmetic
static int instr;                 // a holding register for the current instruction
static int instr_type;                  // the instruction type (opcode)
static int data_loc;                    // the address of the data, or −1 if none
static int data;                  // holds the current operand
static boolean run_bit = true;  // a bit that can be turned off to halt the ma
public static void interpret(int memory[ ], int starting_address) {
  PC = starting_address;
  while (runbit) {
       instr = memory[PC];                          // fetch next instruction into instr
       PC = PC + 1; /                  / increment program counter
       instr_type = get_instr_type(instr);          // determine instruction type
       data_loc = find_data(instr, instr_type);// locate data (−1 if none)
       if (data_loc >= 0)              // if data_loc is −1, there is no operand
                data = memory[data_loc];            // fetch the data
       execute(instr_type, data);      //execute instruction
       }
 }
private static int get_instr_type(int addr) { ... }
private static int find_data(int instr, int type) { ... }
private static void execute(int type, int data){ ... }
}
```


Microprocessor
Data Cache
Memory Bus
RAM
Bus Interface Unit
Control Unit
Arithmetic & Logic Unit
Registers
I/O
System Bus
Instruction Decoder
Floating Point Unit
Registers
Instruction Cache

Lab Equipment

- Development Board: Discovery-IoT (new)
  - Micro-USB Cable (full data cable, not charging)
- Auxiliary material: wires, resistors, buzzer (as per earlier announcement and e-mail)
  - Shopping cart for DigiKey
    - Free shipping if you get some spare parts
- Other tools: multimeter, embedded oscilloscope/logic analyzer
- IDE SW: STM32CubeIDE

IDE


Debug
Project Explorer
try Debug [STM32 Cortex-M C/C++ Application]
try.elf [cores: 0]
Thread #1 [main] 1 [core: 0] (Suspended : Breakpoint
main() at main.c:97 0x8000218
arm-none-eabi-gdb (8.1.0.20180315)
ST-LINK (ST-LINK GDB server)
try.ioc
main.c
{
/* USER CODE BEGIN 1 */
/* USER CODE END 1 */
/* MCU Configuration--*/
/* Reset of all peripherals, Initializes the Flash interface and the Systick. */
HAL_Init();
/* USER CODE BEGIN Init */
/* USER CODE END Init */
/* Configure the system clock */
SystemClock_Config();
/* USER CODE BEGIN SysInit */
/* USER CODE END SysInit */
/* Initialize all configured peripherals */
MX_GPIO_Init();
/* USER CODE BEGIN 2 */
/* USER CODE END 2 */
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14,GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5,GPIO_PIN_RESET);
HAL_Delay(800);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14,GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5,GPIO_PIN_SET);
HAL_Delay(800);
/* USER CODE END WHILE */
/* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}
/**
* @brief System Clock Configuration
* @retval None
*/
Console
Problems
Executables
Debugger Console
Memory
try Debug [STM32 Cortex-M C/C++ Application]
Time elapsed during download operation: 00:00:00.347

Your Jobs Until Next Lecture

- Sign up the lab group online
- Read the documentation
- Try starting STM32CUBE IDE
- Check out a session for the labs (Tue. or Thu.), will be activated after groups are formed, i.e. the week after next one
- Other Preparation: attend tutorials (late next week and afterwards)
- Consult me if there are difficulties