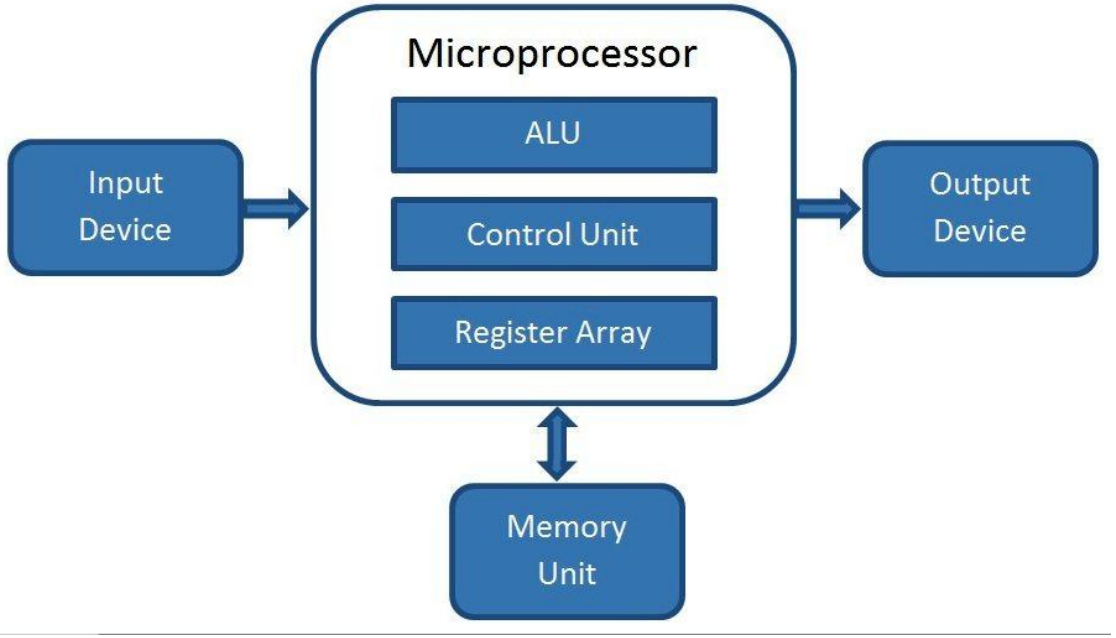
Class # 01

CSE331L: Microprocessor Interfacing & Embedded Systems

Summer 2025

Microprocessor

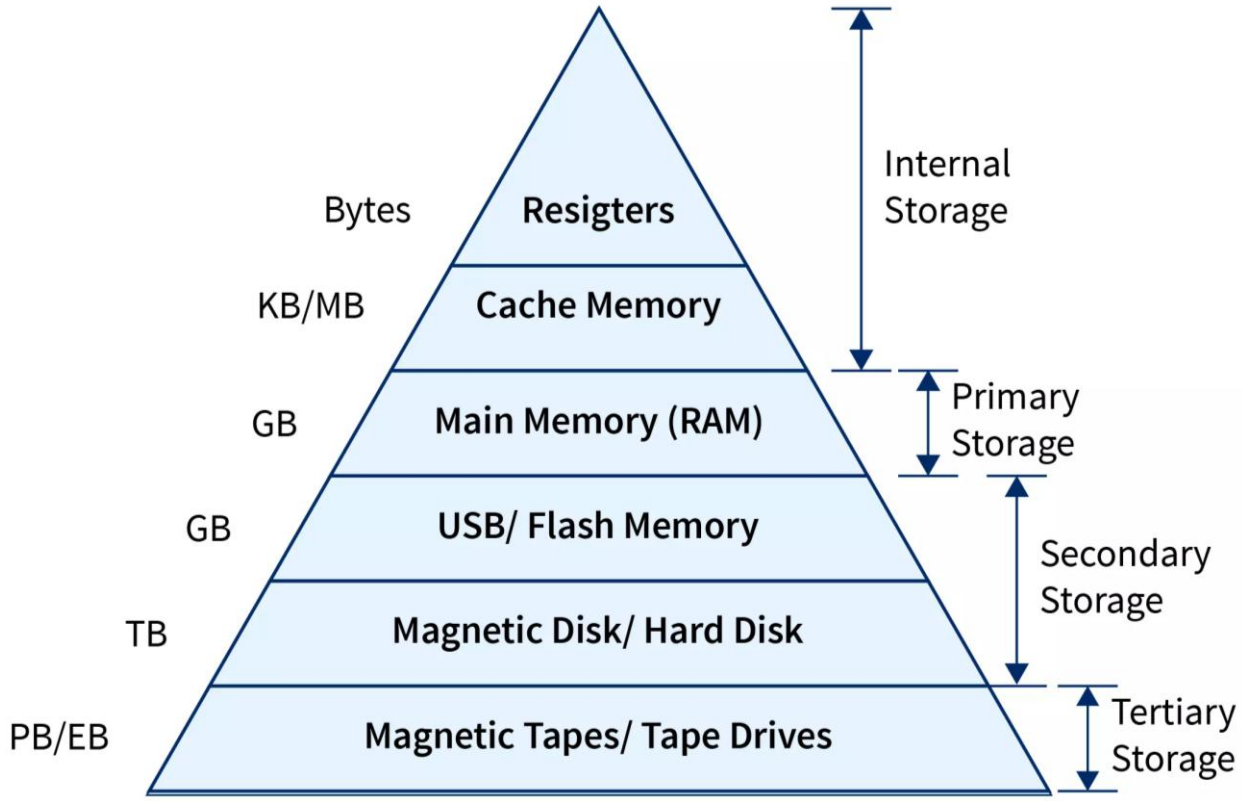
A microprocessor is a tiny computer chip containing a central processing unit (CPU), which is responsible for executing instructions from computer programs



REF

Memory

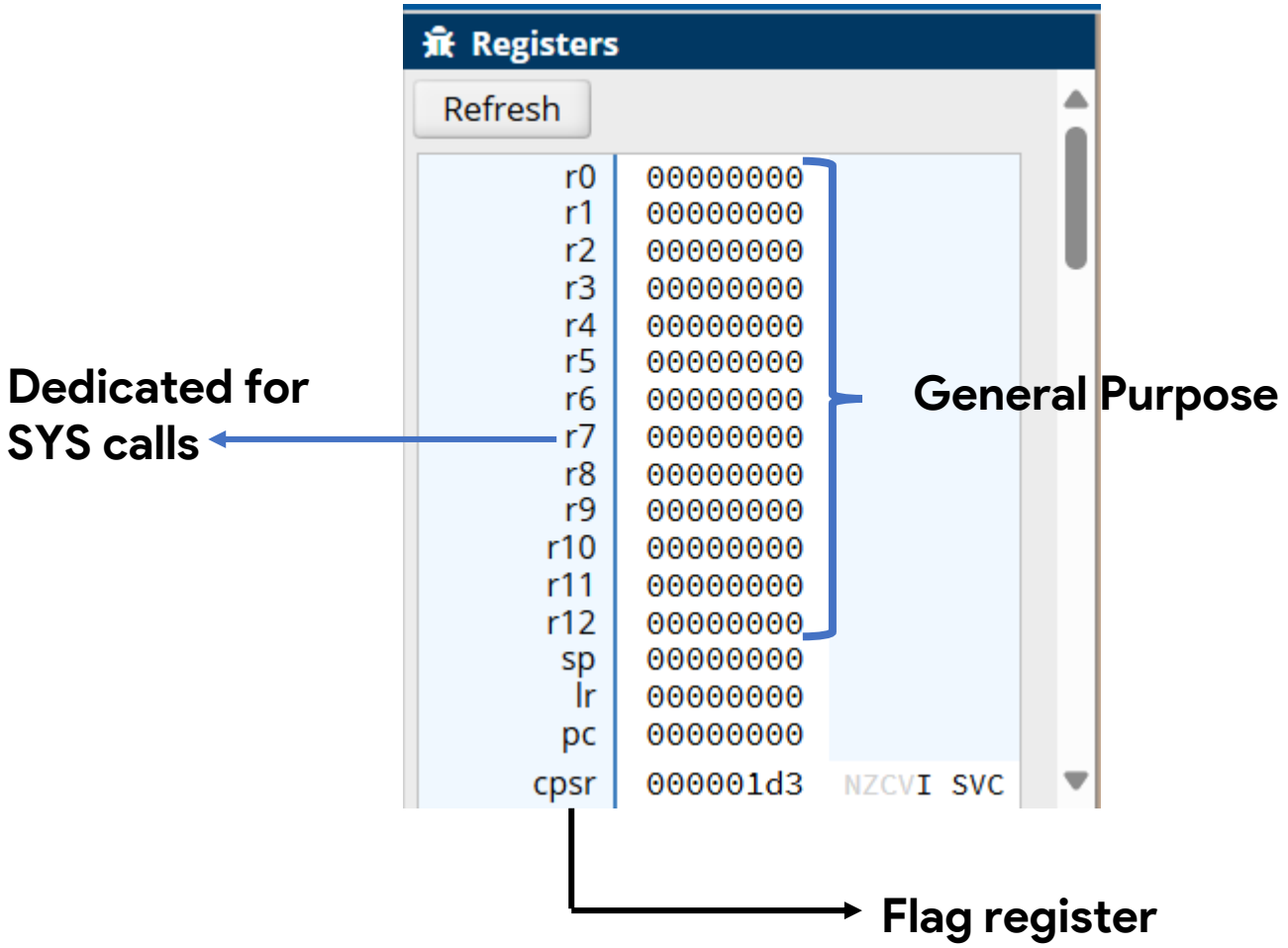
In a computer, memory refers to the storage of data and instructions that the processor needs to access quickly.



REF

Registers

Registers are a type of computer memory built directly into the processor or CPU (Central Processing Unit) that is used to store and manipulate data during the execution of instructions



REF

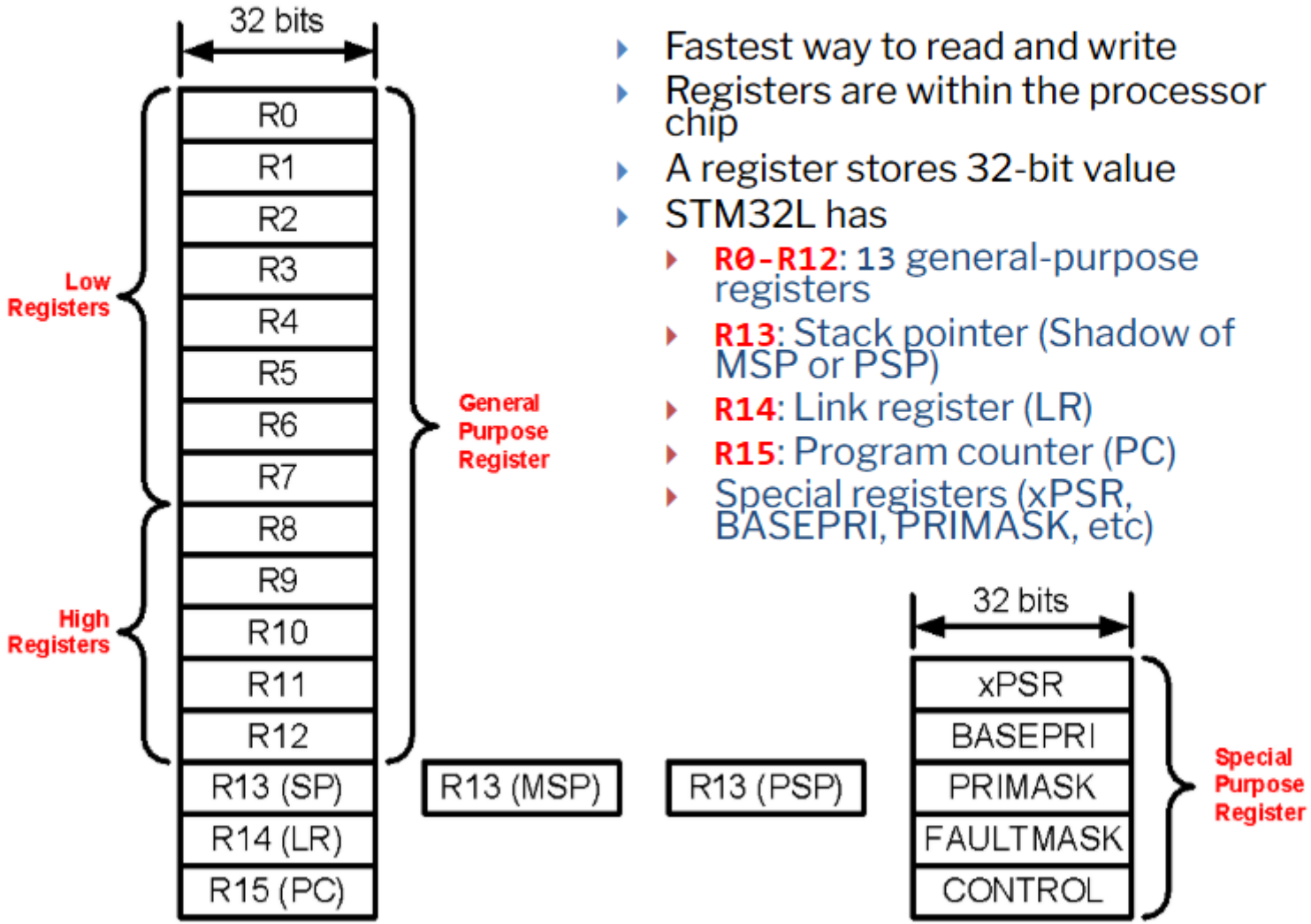
General Purpose Registers

#	Alias	Purpose
R0	-	General purpose
R1	-	General purpose
R2	-	General purpose
R3	-	General purpose
R4	-	General purpose
R5	-	General purpose
R6	-	General purpose
R7	-	Holds Syscall Number
R8	-	General purpose
R9	-	General purpose
R10	-	General purpose
R11	FP	Frame Pointer

General Registers		
R0-R3		Arguments and return values (useable by Thumb16)
R4-R7		General purpose (must be preserved, useable by Thumb16)
R8-R11		General purpose registers (must be preserved)
R12	IP	Intra-procedure-call scratch register
R13	SP	Stack pointer
R14	LR	Return address
R15	PC	Program counter

Reference

Registers–continued



- Fastest way to read and write
- Registers are within the processor chip
- A register stores 32-bit value
- STM32L has
  - **R0-R12**: 13 general-purpose registers
  - **R13**: Stack pointer (Shadow of MSP or PSP)
  - **R14**: Link register (LR)
  - **R15**: Program counter (PC)
  - Special registers (xPSR, BASEPRI, PRIMASK, etc)

Special Purpose Register

Special Purpose Registers		
R12	IP	Intra Procedural Call
R13	SP	Stack Pointer
R14	LR	Link Register
R15	PC	Program Counter
CPSR	–	Current Program Status Register

Reference

ISA (Instruction Set Architecture)

- An Instruction Set Architecture (ISA) is part of the abstract model of a computer that defines how the CPU is controlled by the software.
- The ISA acts as an interface between the hardware and the software, specifying both what the processor is capable of doing as well as how it gets done.

Reference.

RISC

- A Reduced Instruction Set Computer is a type of microprocessor architecture that utilizes a small, highly-optimized set of instructions rather than the highly-specialized set of instructions typically found in other architectures.
- RISC is an alternative to the Complex Instruction Set Computing (CISC) architecture and is often considered the most efficient CPU architecture technology available today.

What is ARM?

ARM is a family of RISC instruction
set **architectures** (ISAs)
for computer processors

CPULator

- For ARM Assembly coding we will be using an simulation website named CPULator.
- Link: https://cpulator.01xz.net/
- We will be using ARMv7 (Architecture)
- Simulation System will be – ARMv7 DE1-SoC (v16.1)
- It is the simulation version of Altera DE1 Board

A code snippet to go through

```
1  .global _start
2  _start:
3      mov r0, #7
4      mov r7, #1
5      add r8, r7, r0
6      SUB r9, r0, r7
7
8      swi 0
9
```

MOV

- Move (register) copies a value from a register to the destination register. It can optionally update the condition flags based on the value.

```
MOV{S}<c> <Rd>, <Rm>
```

Reference

ADD immediate

This instruction adds an immediate value to a register value, and writes the result to the destination register. It can optionally update the condition flags based on the result.

```
ADD{S}<c> <Rd>, <Rn>, #<const>
```

Reference

ADD

- This instruction adds a register value and an optionally-shifted **register value**, and writes the result to the destination register. It can optionally update the condition flags based on the result.

```
ADD{S}<c> <Rd>, <Rn>, <Rm>{, <shift>}
```

Reference

SUB immediate

This instruction subtracts an **immediate value** from a **register value**, and writes the result to the destination register. It can optionally update the condition flags based on the result.

```
SUB{S}<c> <Rd>, <Rn>, #<const>
```

Reference

SUB

- This instruction adds a register value and an optionally-shifted **register value**, and writes the result to the destination register. It can optionally update the condition flags based on the result.

```
SUB{S}<c>.W <Rd>, <Rn>, <Rm>{, <shift>}
```

Reference