MEC302: Embedded Computer Systems

Theme II: Design of Embedded Computer Systems

Lecture 5 – Embedded Processors

Dr. Timur Saifutdinov

Assistant Professor at EEE, SAT

Email: Timur.Saifutdinov@xjtlu.edu.cn

Lecture outline

- What is microprocessor?
- Embedded processors:
    - Microcontrollers;
    - Digital Signal Processors (DSP);
    - Graphics Processing Units (GPU);
- Parallelism and concurrency:
    - Command execution cycle
    - Pipelining
    - Instruction-level parallelism
    - Multicore architectures.

What is microprocessor?

Microprocessor (µP) is a general purpose integrated circuit that processes input data according to instructions from its memory, and provides output. It contains the arithmetic, logic, and control circuitry required to perform the functions of a computer's **Central Processing Unit (CPU)**.

µP realization – piece of silicon manufactured/sold by a semiconductor vendor, eg,:

- Intel/AMD/Cyrix/Atmel/NVIDIA/etc.

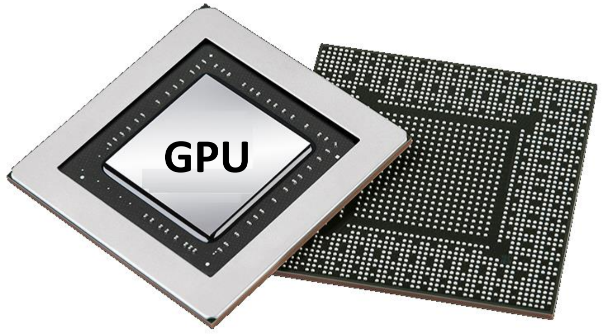
Instruction Set Architecture (ISA) – set of instructions that the **µP** can execute and certain structural constraints (such as word size) that realizations must share, eg:

- **x86(-64)** – dominant in general purpose desktop and laptop computers;
- **ARM** – reduced instruction set computers dominant in smartphones and tablets;
- **8051** – an 8-bit architecture for microcontrollers.
- Many, many, and many more.

Embedded Processors

Main types of embedded processors:

- **Microcontroller (µC)** – a small computer on a single integrated circuit consisting of a relatively simple CPU combined with peripheral devices (i.e., memory, I/O, and timers).
- **Digital Signal Processor (DSP)** – a specifically designed processor to support numerically intensive signal processing applications (e.g., measuring, filtering, and archiving).
- **Graphics Processing Unit (GPU)** – a specialized processor designed especially to perform the calculations required in graphics rendering.



Microprocessors and Microcontrollers

µP is a **CPU** connected to external **peripheral devices**, while **µC** contains them on a single integrated circuit (i.e., chip).

Simple block diagram of a µP [1]:

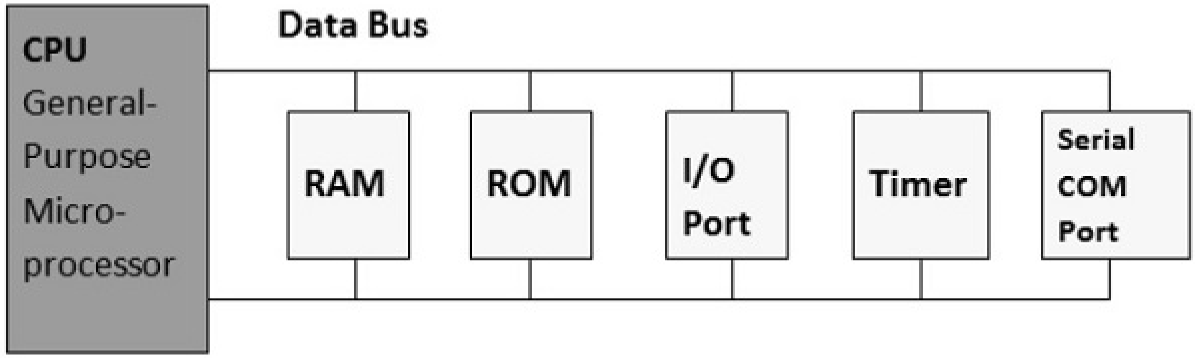
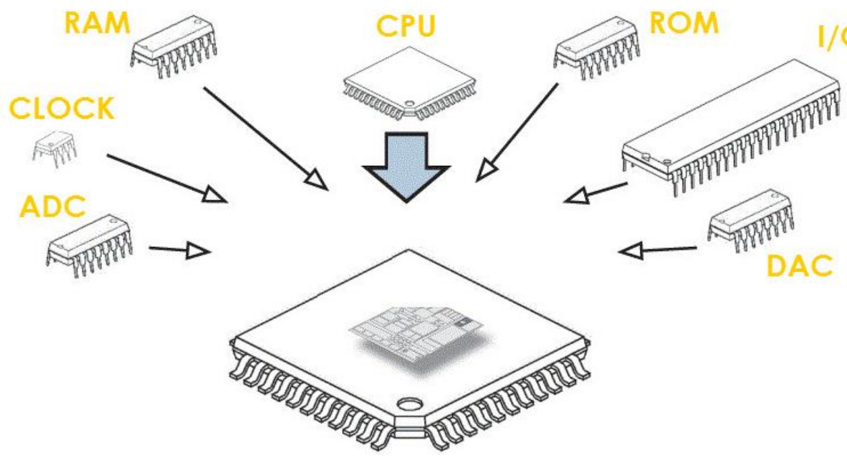


Illustration of a µC [2]:



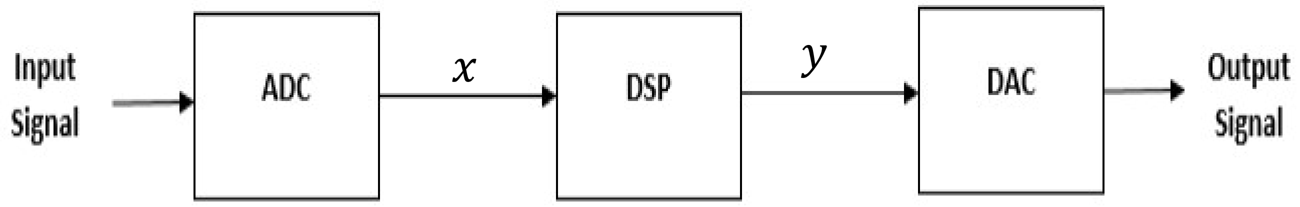
Microprocessor	Microcontroller
Multitasking – can perform multiple tasks	Single task oriented
RAM, ROM, I/O Ports, and Timers can be added	Fixed memory and number of I/Os
Require more space and power	Less demanding on space, weight and power
Expensive	Cheap

[1] Embedded Systems – Processors, https://www.tutorialspoint.com/embedded_systems/es_processors.htm
[2] Basics of Microcontrollers, https://www.electronicshub.org/microcontrollers-basics-structure-applications/

Digital Signal Processor (DSP)

DSP processors are used for processing, filtering and/or compressing digital or analog signals.

Typical DSP system looks as follows [3]:



DSP distinctive features are:

- Process large amounts of data usually at high rate (ie, real-time requirements);
- Perform sophisticated mathematical operations on the data.

DSP applications:

- Speech processing/Image processing/Biometric processing/
- Medical processing/Radars/Seismology/Etc.

[3] What are Different Types of Processors : Applications and Characteristics, https://www.elprocus.com/

Digital Signal Processor (DSP)

DSP processors designed specifically to support numerically intensive signal processing applications.

A canonical signal processing technique is **Finite Impulse Response (FIR)** filtering:

$$y(n) = \sum_{i=0}^{N-1} a_i \, x(n-i)\,,$$

where *n* the current sample, *N* filter length, a_i sample coefficients (i.e., tap values).

#Suppose $N = 4$ and $a_i = 1/4$ then **FIR** filter output is the average of the most recent four input samples:

$$y(n) = \big(x(n) + x(n-1) + x(n-2) + x(n-3)\big)/4.$$

Graphics Processing Unit (GPU)

GPU is a specialized processor designed especially to perform the calculations required in graphics rendering.

Typical GPU workflow [4]:

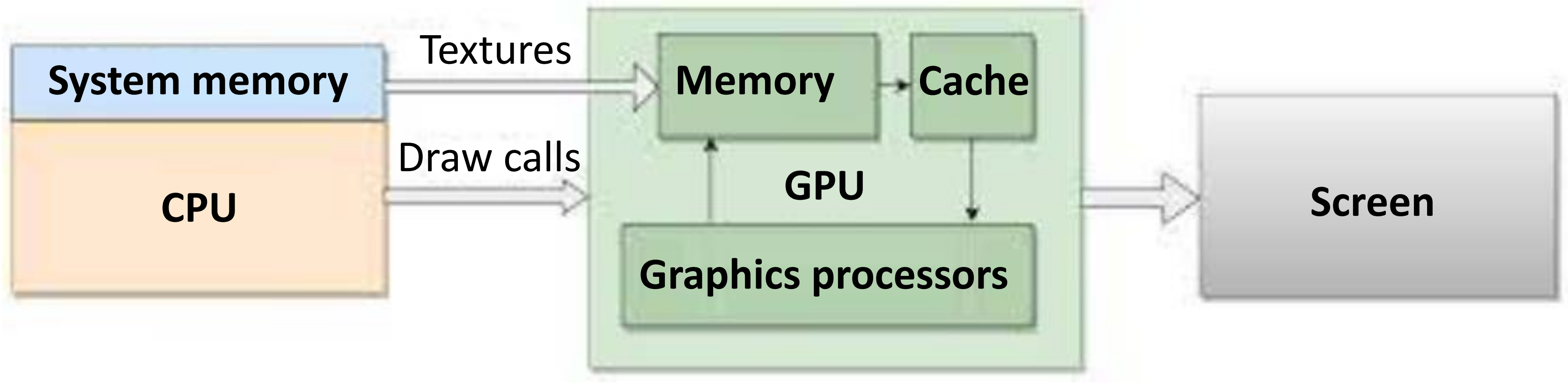
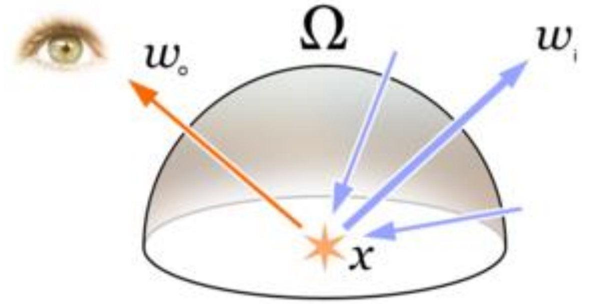


Illustration of the rendering equation [5]:



The key equation in rendering:

$$L_0(\boldsymbol{x},\omega_0,\lambda,t) = L_e(\boldsymbol{x},\omega_0,\lambda,t) + \int_\Omega f_r(\boldsymbol{x},\omega_i,\omega_0,\lambda,t)\, L_i(\boldsymbol{x},\omega_i,\lambda,t)\,(\omega_i\cdot\boldsymbol{n})d\omega_i$$

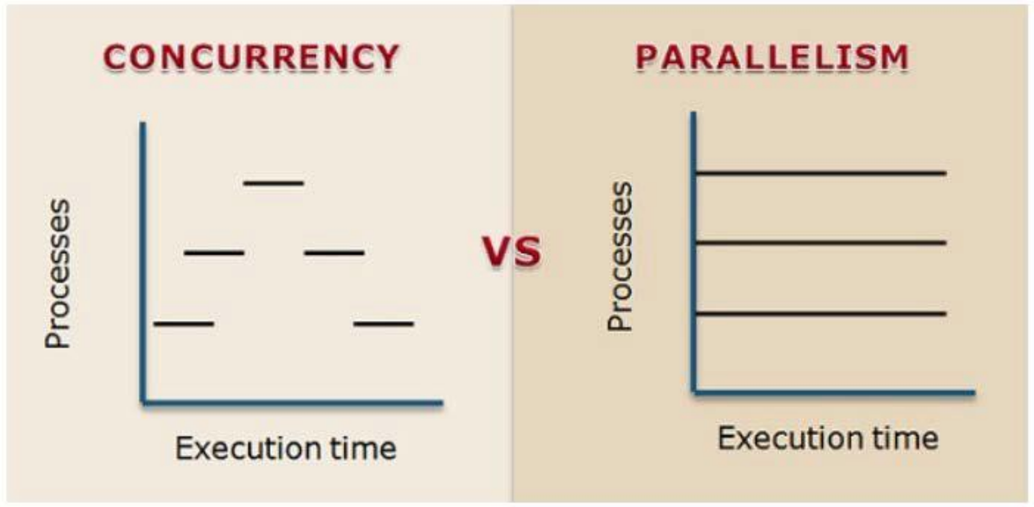
[4] GPU Performance for Game Artists, https://www.gamedeveloper.com/programming/gpu-performance-for-game-artists
[5] Rendering equation, https://en.wikipedia.org/wiki/Rendering_equation

Parallelism and concurrency

In computer systems (i.e., ECS), **parallelism and concurrency** are mechanisms to increase computational efficiency and speed-up execution of programs.

Program commands can be executed:

- Sequentially;
- Concurrently;
- In parallel.

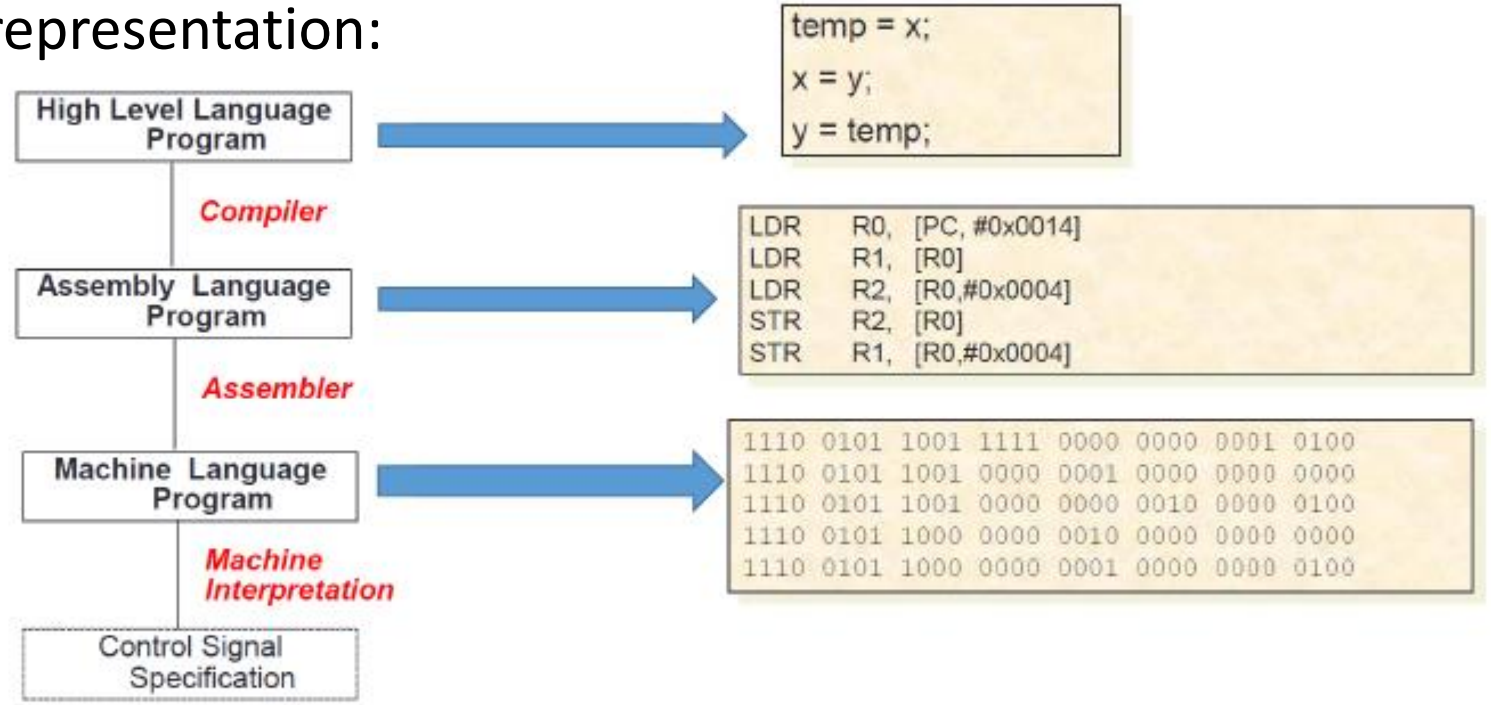


Concurrent program **conceptually** executes different parts of the program simultaneously (e.g., with a single **µP**).

Parallel program **physically** executes different parts of the program simultaneously on distinct hardware (e.g., multiple cores, servers, microprocessors)

Program representations

Levels of representation:


High Level Language Program
Compiler
Assembly Language Program
Assembler
Machine Language Program
Machine Interpretation
Control Signal Specification

```
temp = x;
x = y;
y = temp;
```

```
LDR    R0,  [PC, #0x0014]
LDR    R1,  [R0]
LDR    R2,  [R0,#0x0004]
STR    R2,  [R0]
STR    R1,  [R0,#0x0004]
```

```
1110 0101 1001 1111 0000 0000 0001 0100
1110 0101 1001 0000 0001 0000 0000 0000
1110 0101 1001 0000 0000 0010 0000 0100
1110 0101 1000 0000 0010 0000 0000 0000
1110 0101 1000 0000 0001 0000 0000 0100
```

A programming language that expresses a computation as a sequence of operations is called an **imperative language** (e.g., ***C***).

However, ***C*** allows writing concurrent programs:

- Inexplicitly using **Threads** (thread library);
- Explicitly by **compiler and/or hardware (e.g., pipelining)**.

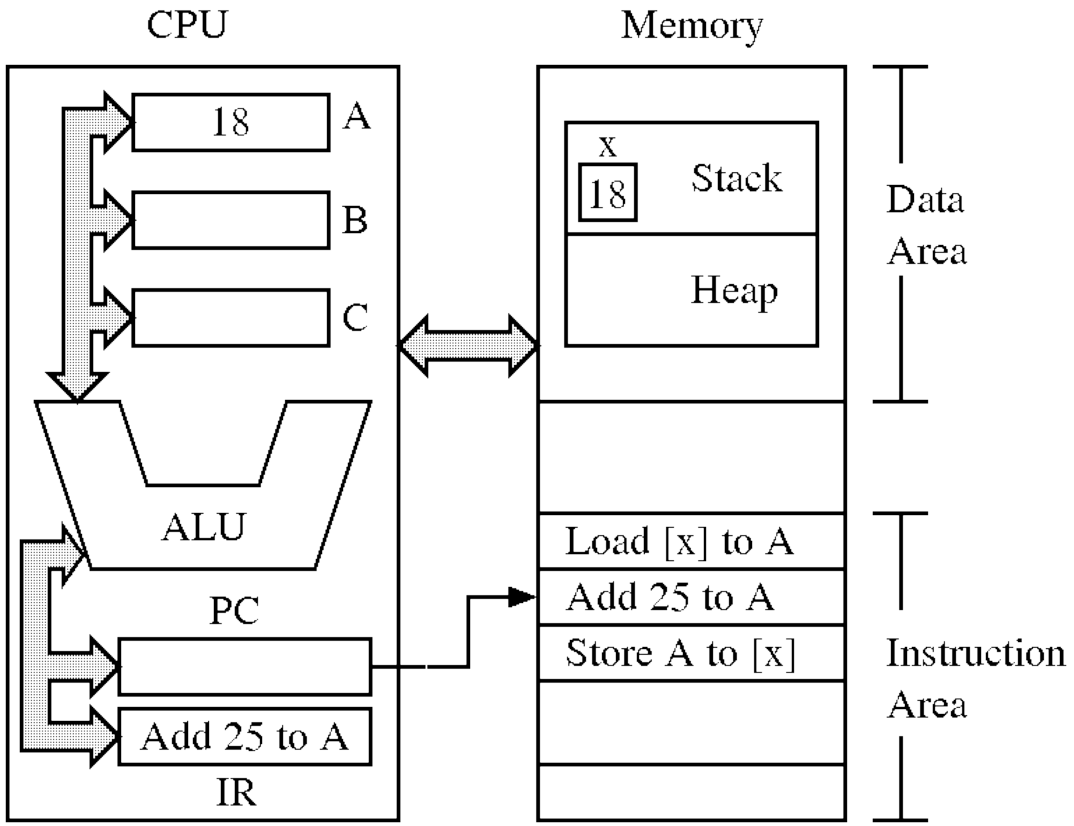
Command execution cycle

A simple instruction (e.g., A:=25+A) requires several tasks to be performed by the CPU (i.e., instruction cycle):

- **Fetch** instruction from the memory;
- **Decode** instruction from the register;
  - And **read** from memory (in case of indirect operation, eg, using variables);
- **Execute** arithmetic/logic operation on data fetched;
  - And **write** to memory;

Proceed to the next command.

Program execution illustration [6].
#Consider the following command:
ADD 25, AddrA, AddrA



[6] Programming and Data Structures, https://cse.iitkgp.ac.in/pds/notes/intro.html

Pipelining

Instruction cycle allows for concurrency using **pipelining**.

#In case of independent commands:

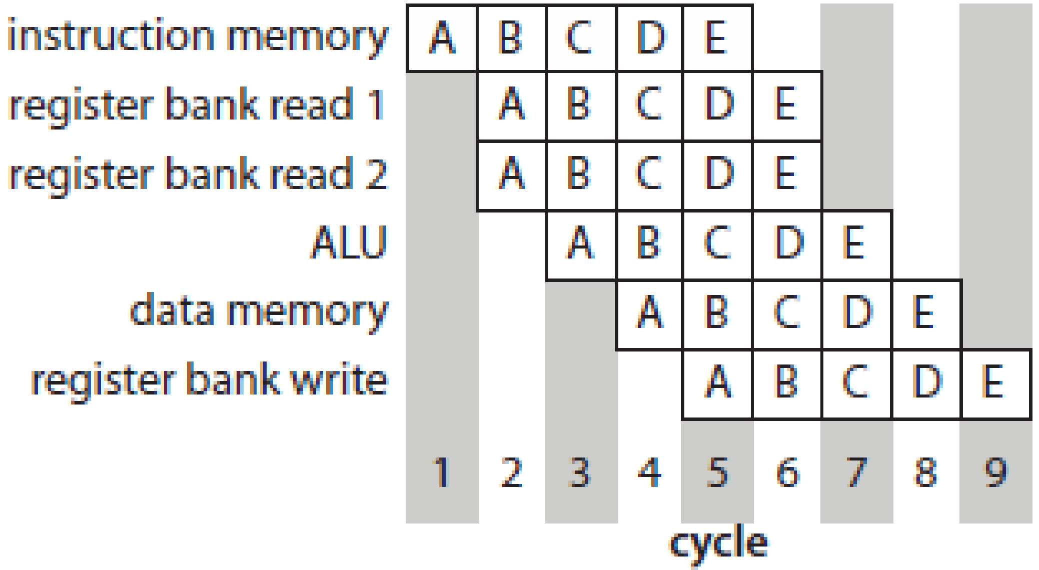
A| a:=b+c;

B| d:=d*e;

C| f:=!g;

…

Concurrent execution of independent tasks:



#In case of dependent commands:

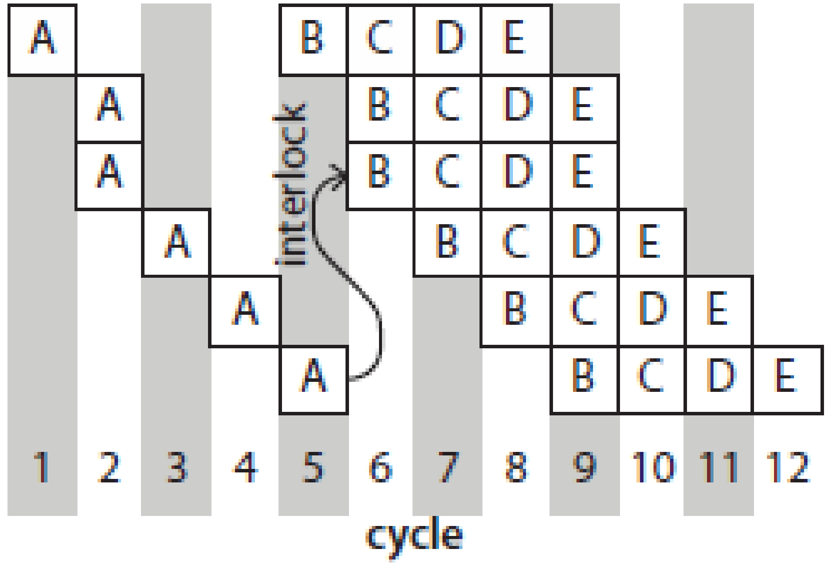
A| a:=b+c;

B| d:=a*e;

C| f:=!g;

…

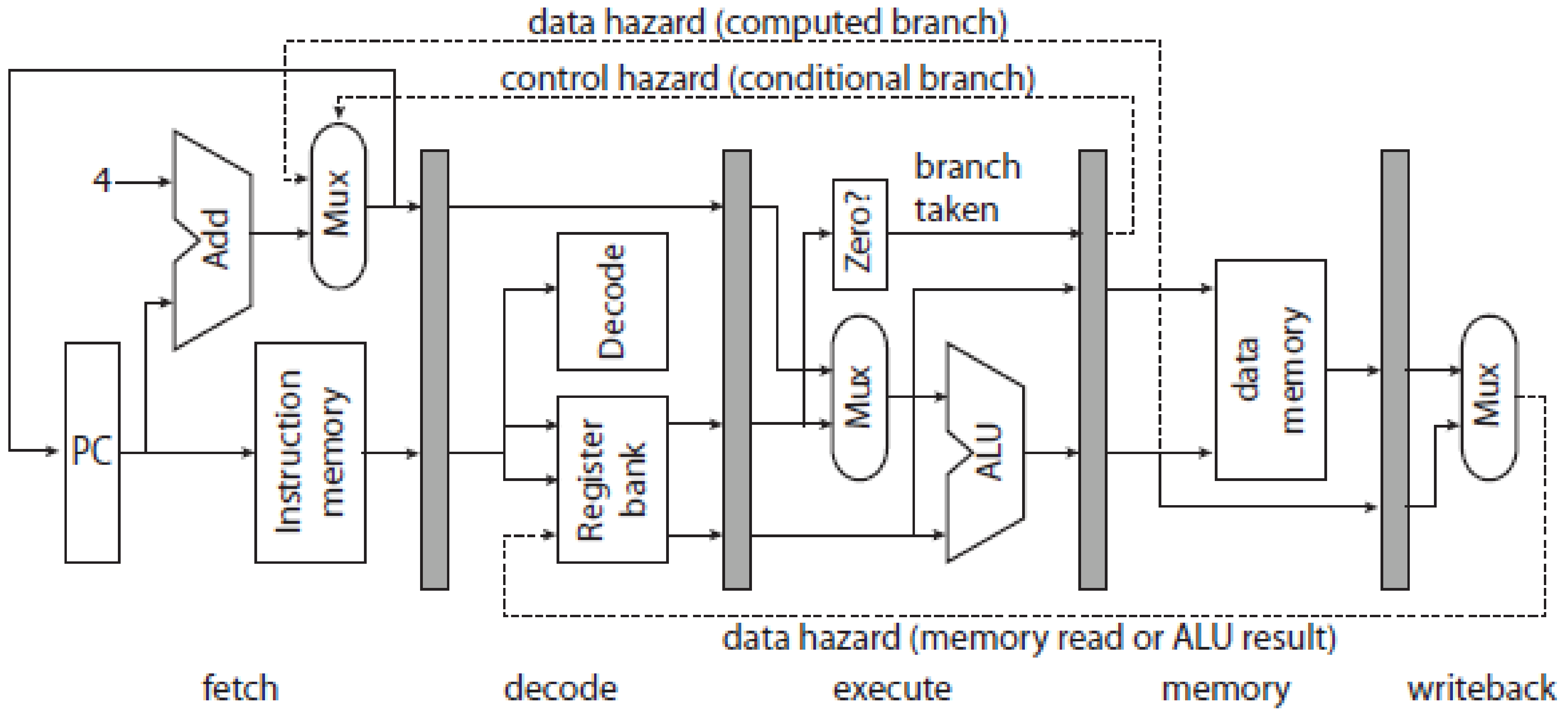
Explicit pipeline implementation (data hazard):



∃ other techniques to facilitate concurrency, for example, **interlocks** and **out-of-order execution**.

Pipelining

#Consider five-stage pipeline for a 32-bit machine [7]:



[7] Patterson, D. A. and J. L. Hennessy, 1996: Computer Architecture: A Quantitative Approach. Morgan Kaufmann, 2nd ed.

Instruction-Level Parallelism (ILP)

A processor supporting **ILP** is able to perform multiple independent operations in each instruction cycle:

- **Complex Instruction Set Computer (CISC)** instructions support specialized complex instructions requiring less cycles (e.g., for FIR filter, fast Fourier transforms and decoding).

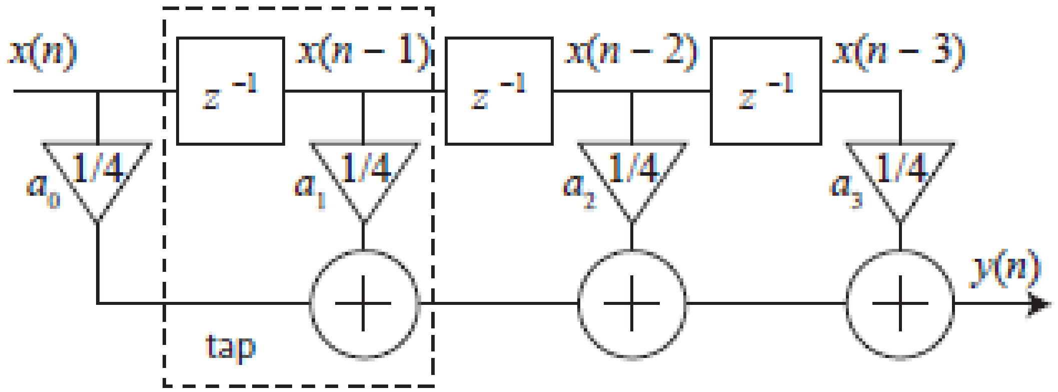
#Consider the CISC DSP realizing FIR filter:

$$y(n) = \sum_{i=0}^{N-1} a_i \, x(n-i)$$

Its ALU can do multiplication and summation in a single operation (a:= a + x*y –one operation):

MAC *AR2+, *AR3+, A

#Consider the actor model of the delay line implementation of the FIR filter for $N = 4$ and $a_i = 1/4$:



Instruction-Level Parallelism (ILP)

Main types of **ILP** for embedded applications:

- **CISC Instructions** – extend standard ALU operations;
- **Subword Parallelism** – enable simultaneous ALU operations on smaller words;
- **Superscalar Processors (SP)** – dispatch multiple instructions to distinct hardware units for simultaneous execution (determined by the processor on-the-fly).
- **Very Large Instruction Word (VLIW)** – similar to SP, where a programmer decides on the dispatch of instructions to separate hardware units.

# Texas Instruments TMS320c55x DSP supports the following commands:

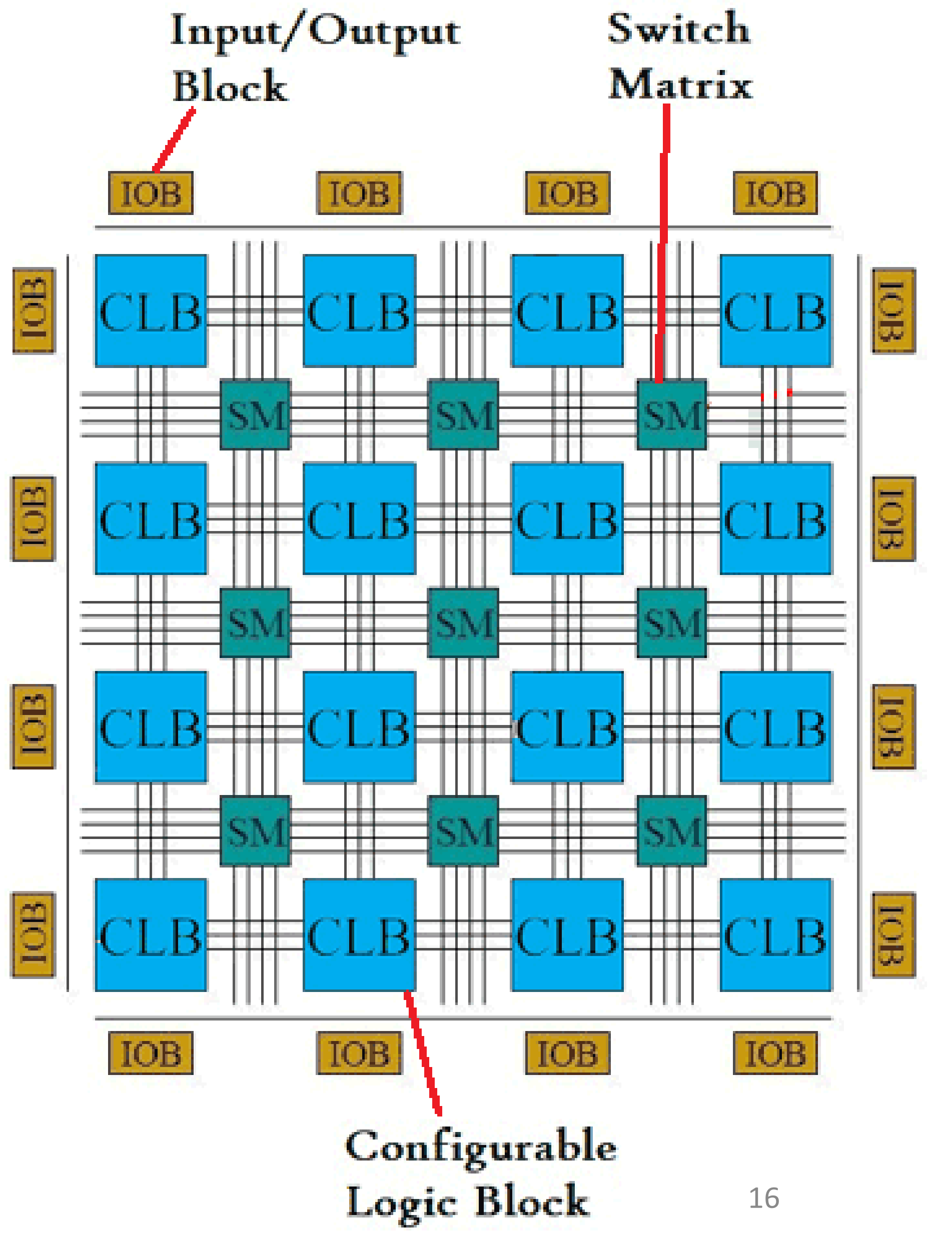
```
MAC     *AR2+,    *CDP+,    AC0
:: MAC  *AR3+,    *CDP+,    AC1
```

:: means that these two instructions should be issued and executed in the same cycle.

Multicore Architectures

To support **SP** and **VLIW**, a **multicore machines** are required – multiple **CPUs** on a single chip. There are:

- **Homogenous multicore machines** combine a number of processors of the same type on a single chip.
- **Heterogeneous multicore machines** combine a variety of processor types on a single chip.
- **Field-Programmable Gate Array** (**FPGA**) is the type of the multicore architecture that is used in embedded applications – uses one or more soft cores together with custom hardware.



To sum up

- **Microprocessor (μP)** is a central part of the computer systems that perform the functions of CPU;
- Processors for ECS designed cognizant of their intended applications:
    - **Microcontroller (μC)** – general-purpose and application specific **μP**;
    - **Digital Signal Processors (DSP)** – specialized **μP** for signal processing (e.g., filtering or archiving);
    - **Graphics Processing Units (GPU)** – specialized **μP** for graphics rendering;
- **Parallelism** and **concurrency** are powerful mechanisms to increase computational efficiency and speed-up execution of programs:
    - **Command execution cycle** allows for concurrency (i.e., pipelining) even with a single **μP**;
    - **Instruction-Level Parallelism (ILP)** require **μP** to implement complex instructions;
    - **Parallelism** uses distinct hardware (i.e., **μPs**) for simultaneous execution of programs.

The end!

See you next time (March 27)