



NPTEL ONLINE CERTIFICATION COURSES

Course Name: Embedded System Design with ARM

Faculty Name: Prof. Indranil Sen Gupta

Department : Computer Science and Engineering

Topic

Lecture 1: Introduction to Embedded Systems

CONCEPTS COVERED

- ☐ Introduction to embedded systems
- ☐ Applications of embedded systems
- ☐ Typical subsystems in an embedded system



Introduction

- We have been brought up in the age of computing.
 - Computers are everywhere (some we see, some we do not see).
- Types of computers we are familiar with:
 - Desktops and Laptops
 - Servers
 - Mobile phones
- But there's another type of computing system that is often hidden.
 - Far more common and pervasive...
 - Hidden in the environment.

Embedded Systems

What are Embedded Systems?

- Computers are embedded within other systems:
 - What is “*other systems*”? – Hard to define.
 - Any computing system other than desktop / laptop server.
 - Typical examples:
 - Washing machine, refrigerator, camera, vehicles, airplane, missile, printer.
 - Processors are often very simple and inexpensive (depending on application of course).
- Billions of embedded system units produced yearly, versus millions of desktop units.



Common Features of Embedded Systems

- They are special-purpose or single-functioned.
 - Executes a single program, possibly with inputs from the environment.
 - Imagine a microwave oven, a washing machine, an AC machine, etc.
- Tight constraints on cost, energy, form factor, etc.
 - Low cost, low power, small size, relatively fast.
- They must react to events in real-time.
 - Responds to inputs from the system's environment.
 - Must compute certain results in real-time without delay.
 - The delay that can be tolerated depends on the application.

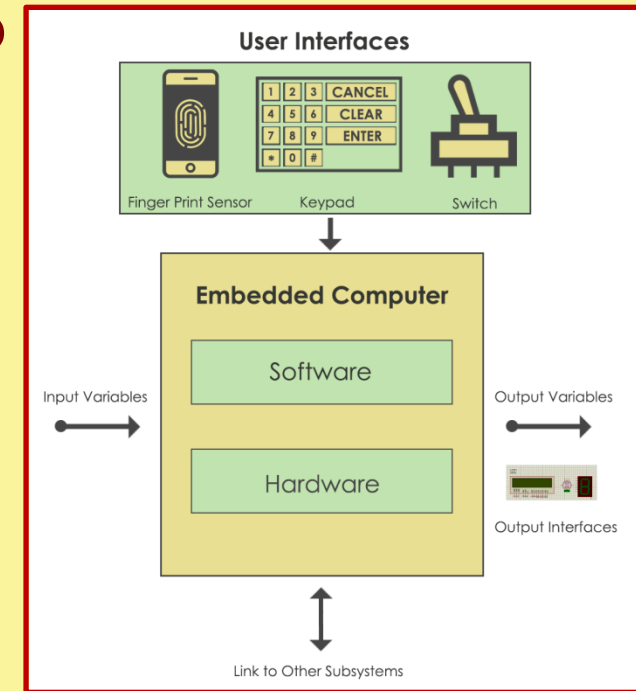


Typical Design Constraints

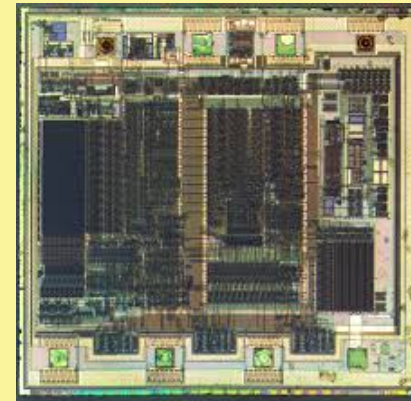
- Low Cost
 - A sophisticated processor can increase the cost of the embedded system.
- Low Energy Consumption
 - Many embedded systems operate on battery.
- Limited Memory
 - Typically constrained to a finite and small amount of memory.
- Real-Time Response
 - Most embedded systems are used for controlling some equipment.
 - Must generate response within a specified time.

How to define an Embedded System?

- It is a microcontroller-based system that is designed to control a function or range of functions, and is not meant to be programmed by the end user.
 - The user may make choices concerning the functionality but cannot change them.
 - The user cannot make modifications to the software.
 - Can you “*program*” your washing machine or refrigerator or car?
 - Not today ... but not very sure of the near future.



- What embedded system is not ...
 - A microprocessor sitting inside a traditional computing system (like desktop, laptop, server, etc.).
- It is actually:
 - A microprocessor used to control another piece of technology (dedicated, and not general-purpose).
 - For low cost, microcontrollers that are typically used are single-chip devices containing processor, memory, and I/O interfaces.

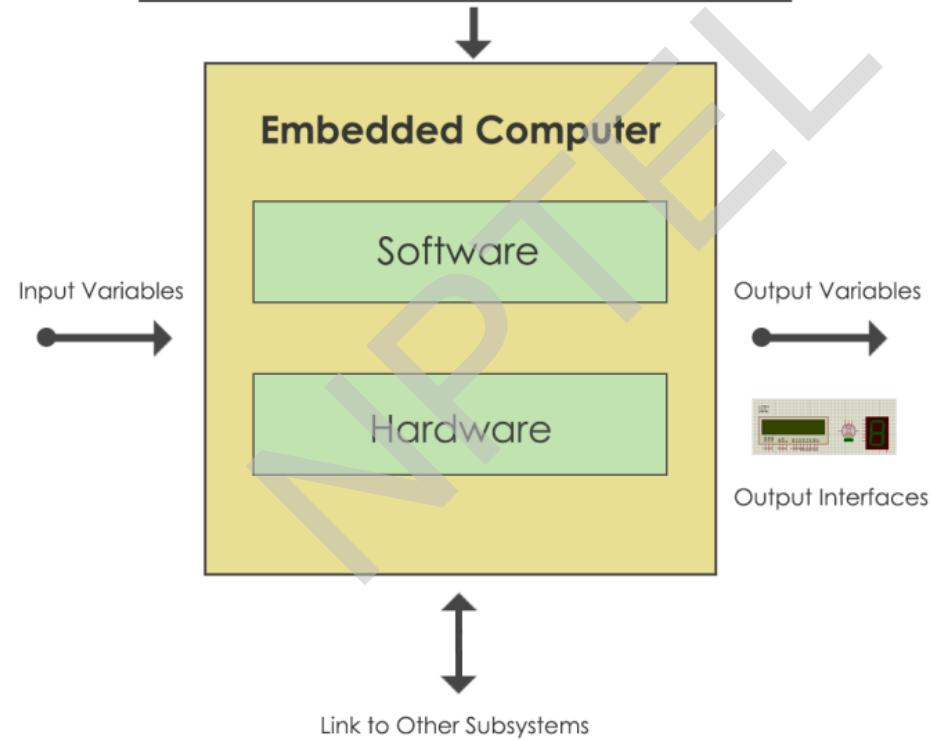
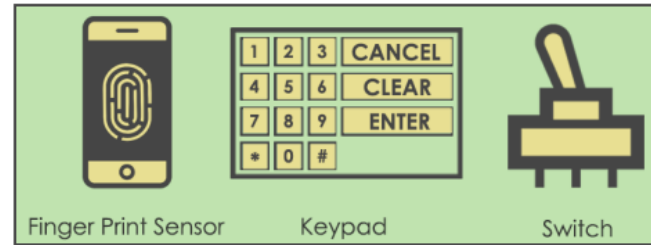


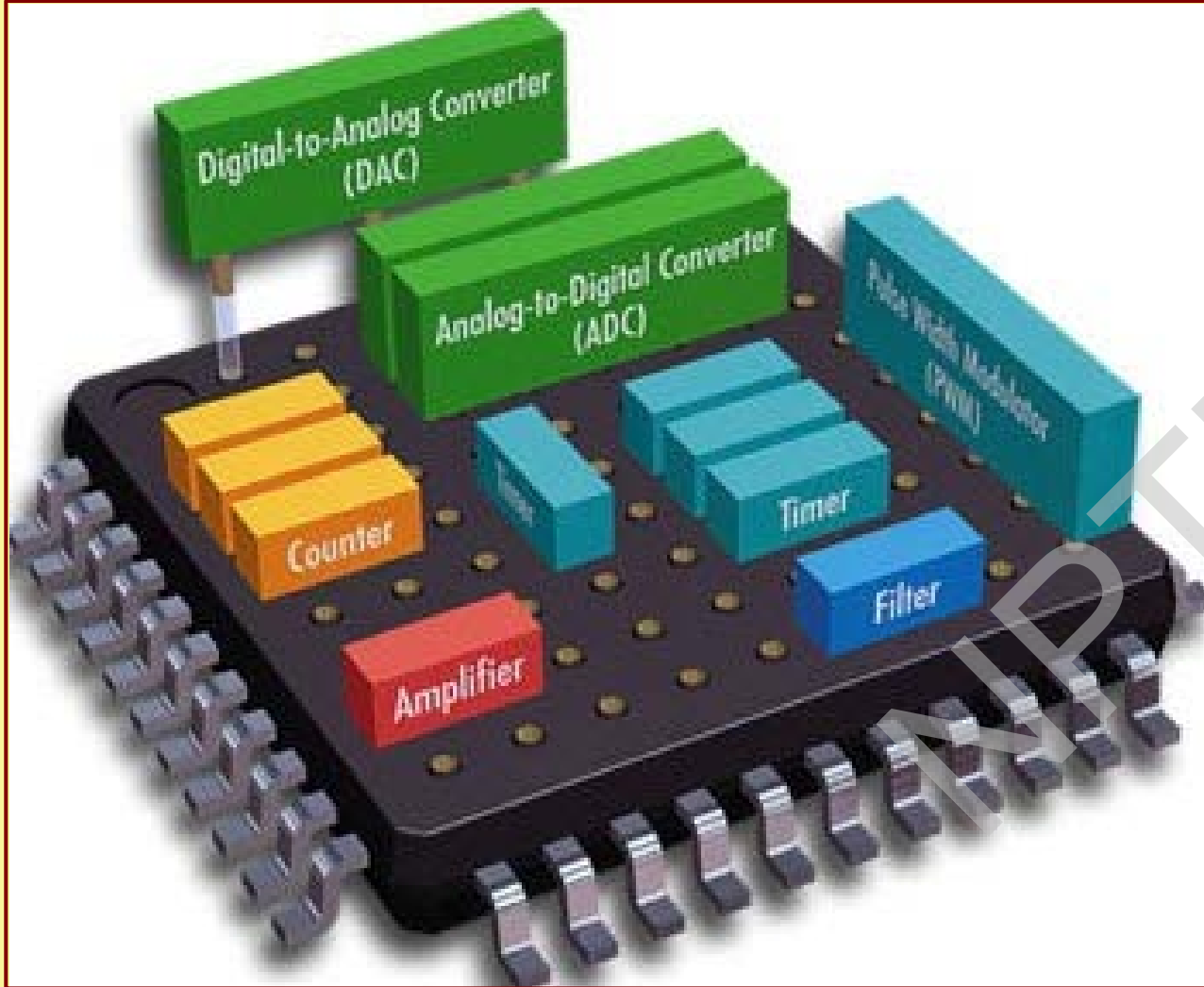
Applications of Embedded Systems

- *Limited by imagination.*

- a) **Consumer Segment:** Refrigerator, washing machine, A/C machine, camera, microwave oven, TV, security system, etc.
- b) **Office Automation:** Printers, Fax machines, photocopying machines, scanners, biometric scanner, surveillance camera, etc.
- c) **Automobiles:** Air bags, anti-lock braking system (ABS), engine control, door lock, GPS system, vehicular ad-hoc network (VANET), etc.
- d) **Communication:** Mobile phones, network switches, WiFi hotspots, telephones, MODEM, etc.
- e) **Miscellaneous:** Automatic door locks, automatic baggage screening, surveillance systems, intelligent toilet, etc.

User Interfaces





Notable subsystems:

- a) Analog-to-digital (ADC) interfaces
- b) Digital-to-analog (DAC) interfaces
- c) Pulse-width-modulation (PWM) interfaces
- d) Timers and counters
- e) In addition to ... processor, memory, digital I/O ports, etc.



NPTEL ONLINE CERTIFICATION COURSES

Thank
you!



NPTEL ONLINE CERTIFICATION COURSES

Course Name: Embedded System Design with ARM

Faculty Name: Prof. Indranil Sen Gupta

Department : Computer Science and Engineering

Topic

Lecture 2: Design Considerations of Embedded Systems

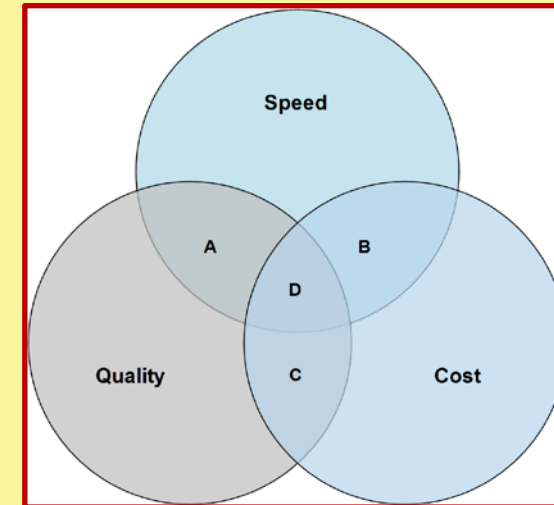
CONCEPTS COVERED

- ☐ Design challenges for embedded systems
- ☐ Understanding design tradeoffs
- ☐ Non-recurring cost and unit cost metrics



Design Challenges

- Primary design goal:
 - An implementation that realizes the desired functionality.
- The main design challenge is ...
 - To simultaneously optimize several design metrics.
 - Often mutually conflicting.
- What is a design metric?
 - Some feature of an implementation that can be *measured* and *evaluated*.



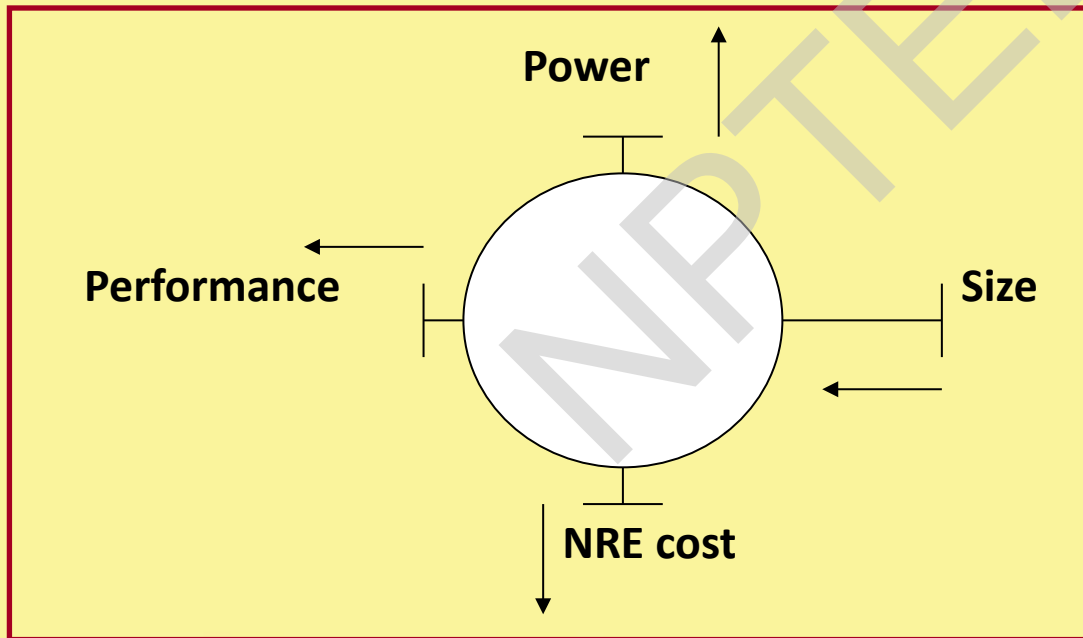
Common Design Metrics

- **Non Recurring Engineering (NRE) Cost:** One-time initial cost of designing a system.
- **Unit Cost:** The cost of manufacturing each copy of the system, without counting the NRE cost.
- **Size:** The actual physical space occupied by the system.
- **Performance:** This is measured in terms of the time taken or throughput.
- **Power:** The amount of (battery) power consumed by the system.
- **Flexibility:** The ability to change the functionality of the system.

- **Maintainability**: How easy or difficult it is to modify the design of the system?
- **Time-to-prototype**: How much time is required to build a working version of the system (i.e. a prototype)?
- **Time-to-market**: How much time is required to develop a system such that it can be released to the market commercially?
- **Safety**: Are there any adverse effects on the operating environment?
- Can be many more ...

Design Tradeoff

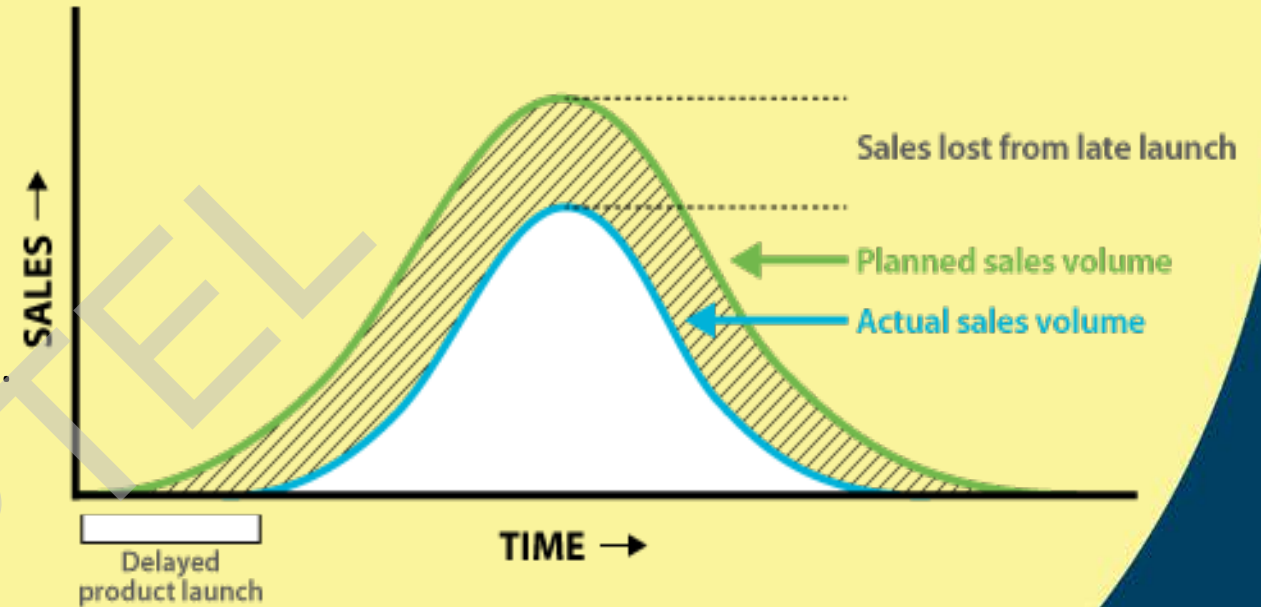
- Many of the design metrics can be mutually conflicting.
 - Improving one may degrade the other (e.g. power, performance, size and NRE cost, etc.).



- Often requires expertise in both hardware and software to take a proper decision.
 - Expertise in hardware may indicate the types of co-processor or I/O interfaces to use for specific applications (e.g. analog ports, digital ports, PWM ports, etc.).
 - Expertise in software is required to identify parts of the implementation that need to be implemented in software and run on the microcontroller.
 - Hardware / Software Co-design becomes important.

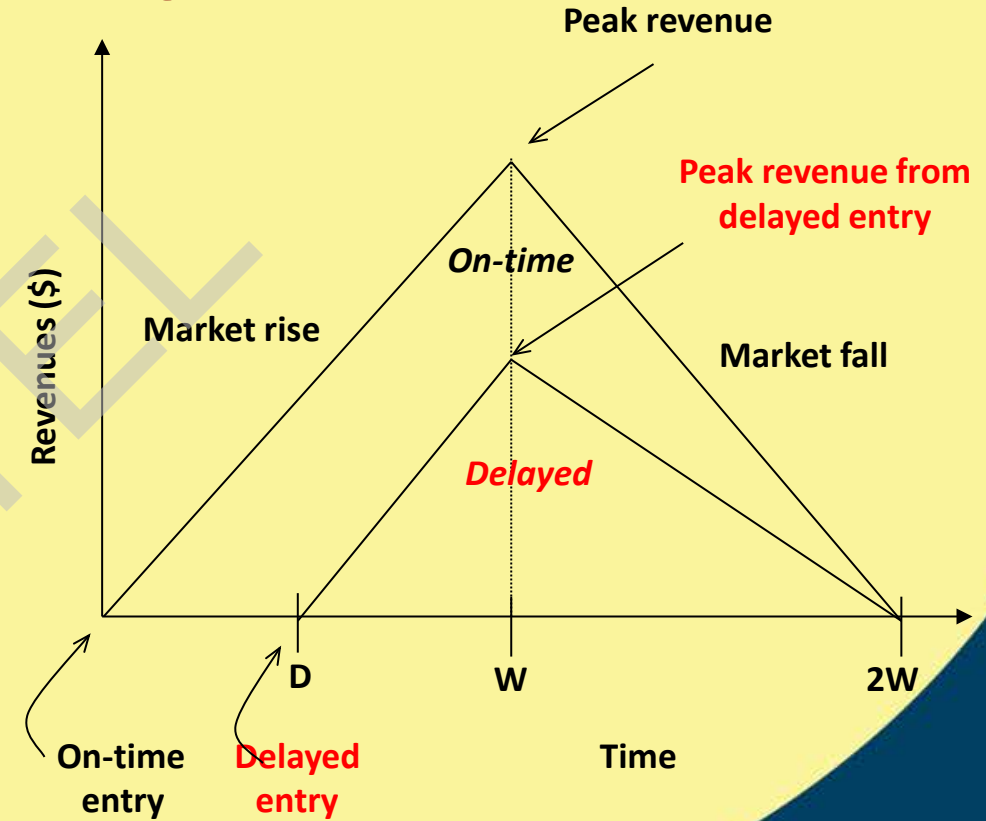
Time-to-market Design Metric

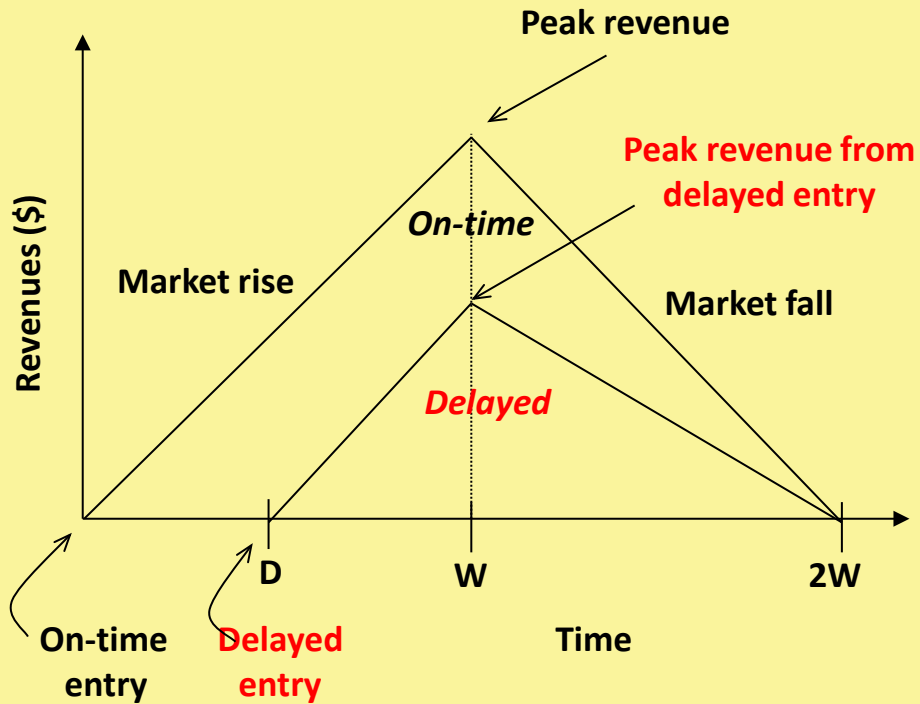
- This is a very crucial design metric.
 - Must be strictly followed to make a product commercially viable.
 - Requires exhaustive market study and analysis.
- Starting from the point a product design starts, we can define a *Market Window* within which it is expected to have the highest sales.
 - Any delay can result in drastic reductions in sales.



Loss due to Delayed Market Entry

- Consider a simplified revenue model:
 - Product life is $2W$
 - Maximum sale occurs at time W
 - Market rise and market fall defines a *triangle* in the revenue graph.
 - Area of the triangle determines the total revenue.*
- For delayed entry, can estimate the loss.
 - Difference in areas of the expected and actual triangles.





- Area of a triangle = $\frac{1}{2} * \text{base} * \text{height}$
 - Area (on-time) = $\frac{1}{2} * 2W * W$
 - Area (delayed) = $\frac{1}{2} * (2W - D) * (W - D)$
 - Percentage revenue loss = $D(3W - D)/2W^2 * 100$
- Examples:
 - $2W = 52$ weeks, $D = 4$ weeks $\rightarrow$ LOSS = 22%
 - $2W = 52$ weeks, $D = 10$ weeks $\rightarrow$ LOSS = 50%

NRE and Unit Cost Metrics

- If C_{NRE} denotes the NRE cost and C_{unit} the unit cost of a product, then the total cost for manufacturing N units is given by:

$$Total\ Cost = C_{NRE} + N * C_{unit}$$

- Therefore, per-unit cost is given by: $C_{NRE} / N + C_{unit}$
- Example:
 - C_{NRE} = Rs. 5,00,000 and C_{unit} = Rs. 5,000
 - Total cost for manufacturing 100 units = 5,00,000 + 5000 * 100 = 10,00,000
 - Per unit cost = 5,00,000 / 100 + 5000 = 10,000

$$\text{Per-unit cost} = C_{NRE} / N + C_{unit}$$

- We can compare technologies by cost:
 - Choice A: $C_{NRE} = \text{Rs. } 20,000$, $C_{unit} = \text{Rs. } 8,000$
 - Choice B: $C_{NRE} = \text{Rs. } 4,00,000$, $C_{unit} = \text{Rs. } 3,000$
 - Choice C: $C_{NRE} = \text{Rs. } 10,00,000$, $C_{unit} = \text{Rs. } 8,000$
- Of course, time-to-market cost must also be considered.

Performance Design Metric

- Most widely used, but can also be most misleading.
 - Must be careful in the evaluation.
- Some of the measures:
 - Clock frequency, MIPS --- not very good for comparison
 - Latency (response time)
 - Throughput
- Measure of speedup among design alternatives.



NPTEL ONLINE CERTIFICATION COURSES

**Thank
you!**



NPTEL ONLINE CERTIFICATION COURSES

Course Name: Embedded System Design with ARM

Faculty Name: Prof. Indranil Sen Gupta

Department : Computer Science and Engineering

Topic

Lecture 3: Microprocessors and Microcontrollers

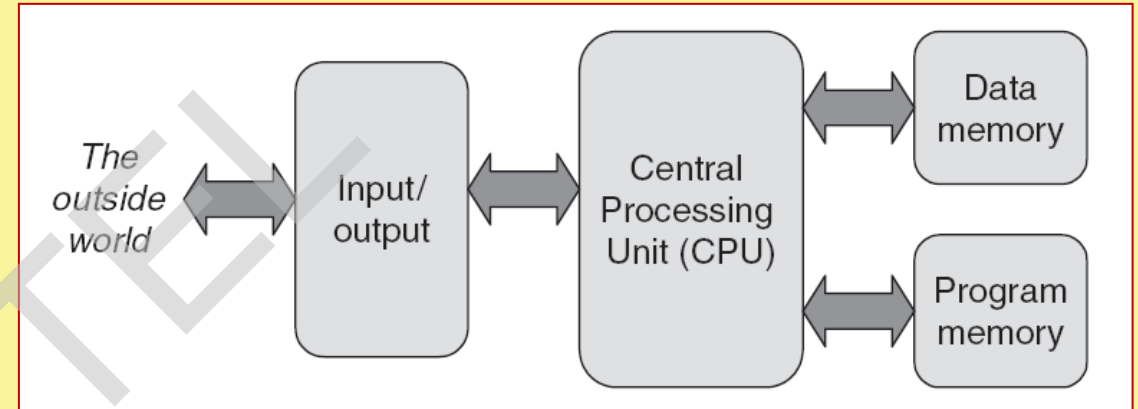
CONCEPTS COVERED

- ☐ Classification of computer architecture
- ☐ Characteristics of a microprocessor
- ☐ Characteristics of a microcontroller



Basic Operation of a Computing System

- The central processing unit (CPU) carries out all computations.
 - Fetches instructions from the program memory and executes it; may require access to data in data memory.
- The input/output block provides interface with the outside world.
 - Allows users to interact with the computing system, and also observe the output results.

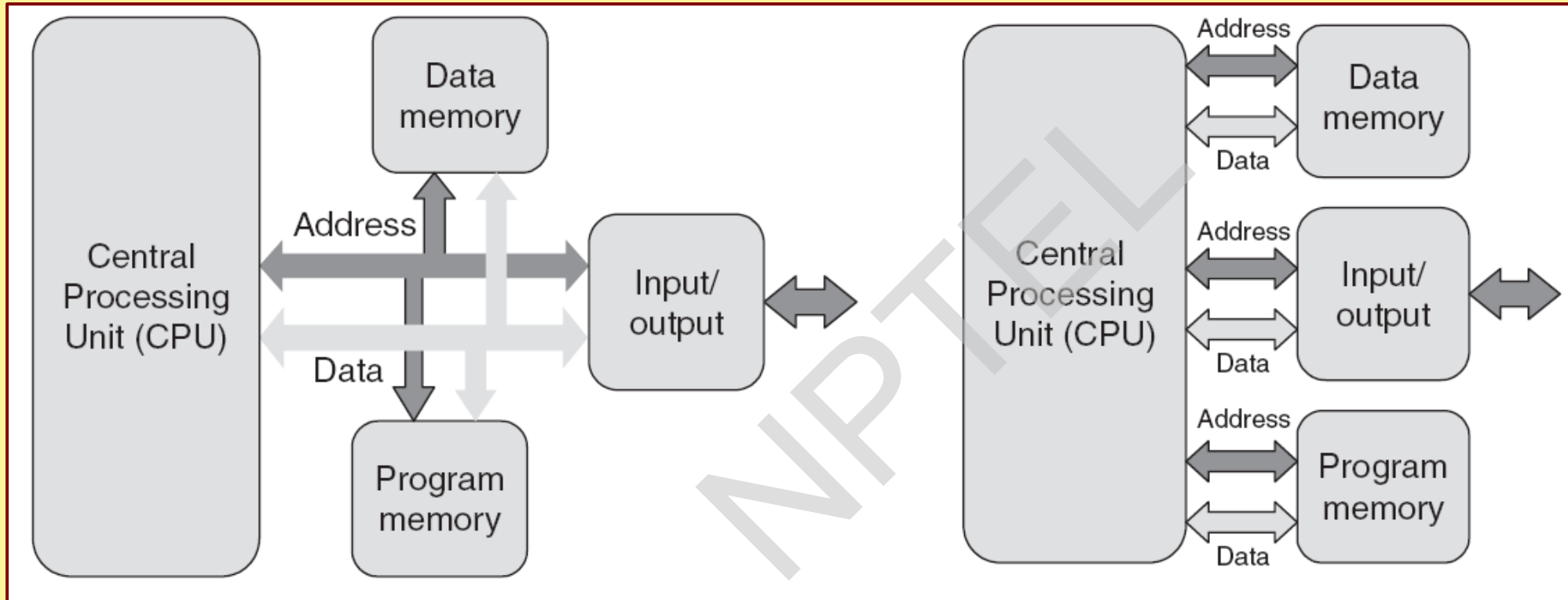


- About the instruction set architecture (ISA) of the CPU.
 - a) Complex Instruction Set Computer (CISC)
 - Typically used in desktops, laptops and servers (courtesy Intel).
 - b) Reduced Instruction Set Computer (RISC)
 - Typically used in microcontrollers, that are used to build embedded systems.
- Two different types of memory:
 - a) Random Access Memory (RAM)
 - Volatile; used for data memory in microcontrollers.
 - b) Read Only Memory (ROM)
 - Non-volatile; used for program memory in microcontrollers.

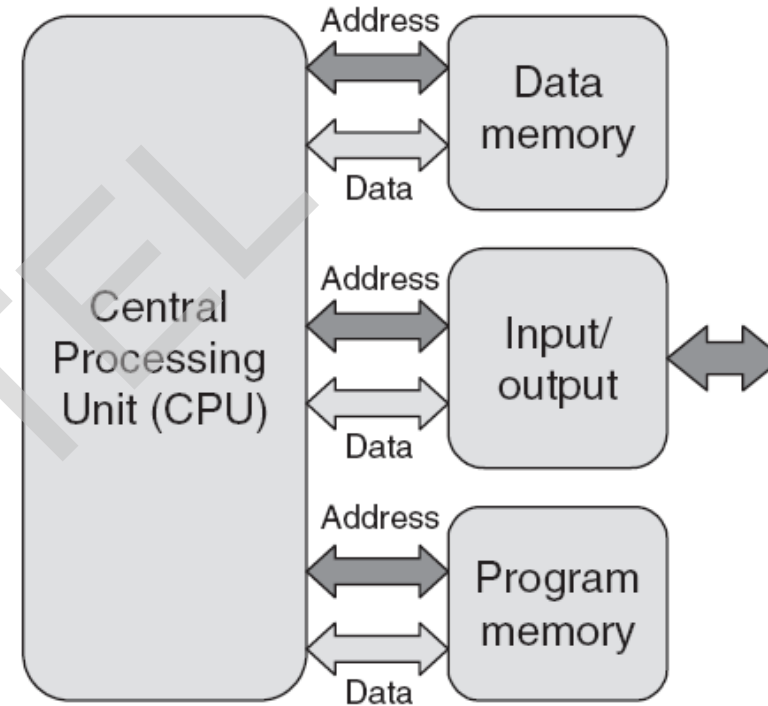
Classification of CPU Architecture

- Broadly two types of architectures:
 - a) Von Neumann Architecture
 - Both instructions and data are stored in the same memory.
 - This model is followed in conventional computing systems.
 - b) Harvard Architecture
 - Instructions and data are stored in separate memories.
 - Typically followed in microcontrollers, used for building embedded systems.
 - Instructions are stored in a ROM (permanent), while temporary data are stored in RAM.

Von Neumann Architecture



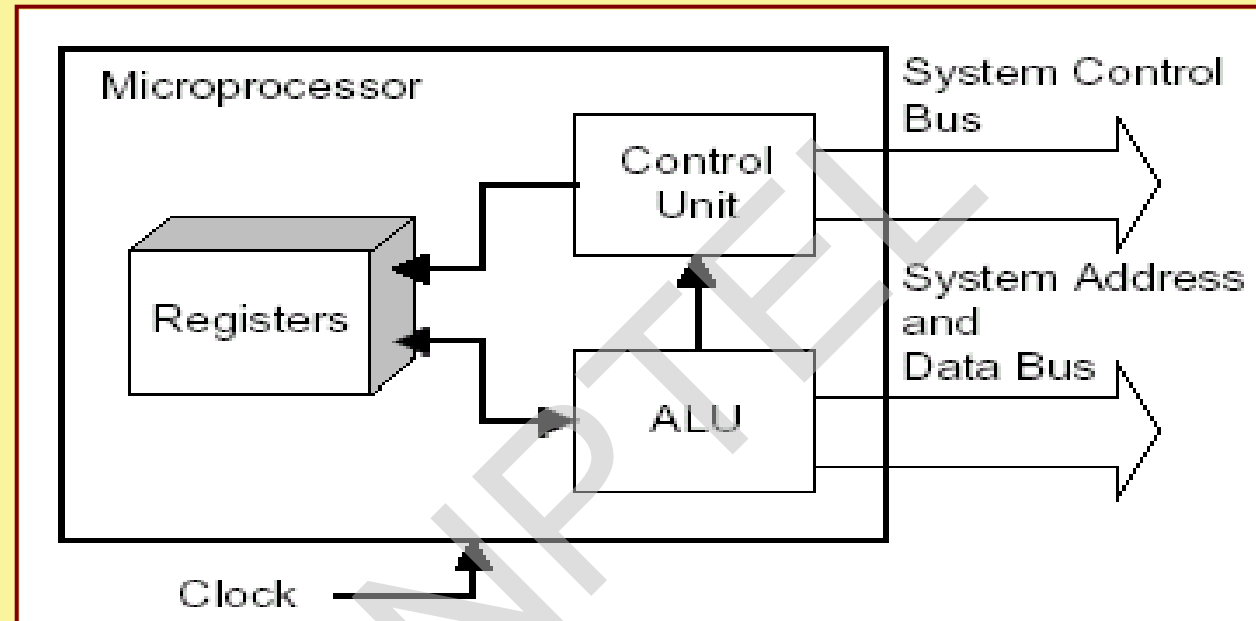
Harvard Architecture



What is a Microprocessor?

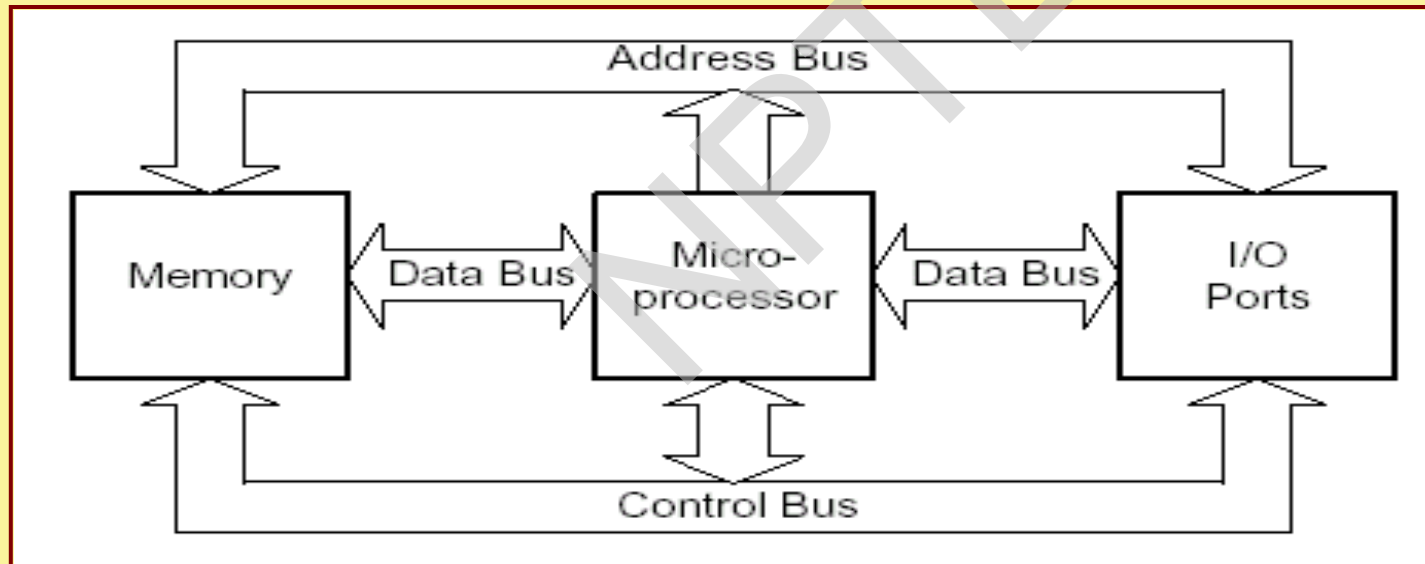
- It is basically the entire CPU fabricated on a single chip.
 - Consists of a set of registers to store temporary data.
 - Consists of an arithmetic logic unit (ALU), where all arithmetic and logical computations are carried out.
 - Consists of some mechanism to interface external devices (memory and I/O) through buses (address, data and control).
 - Consists of a control unit that synchronizes the operation.

Schematic Diagram of Microprocessor



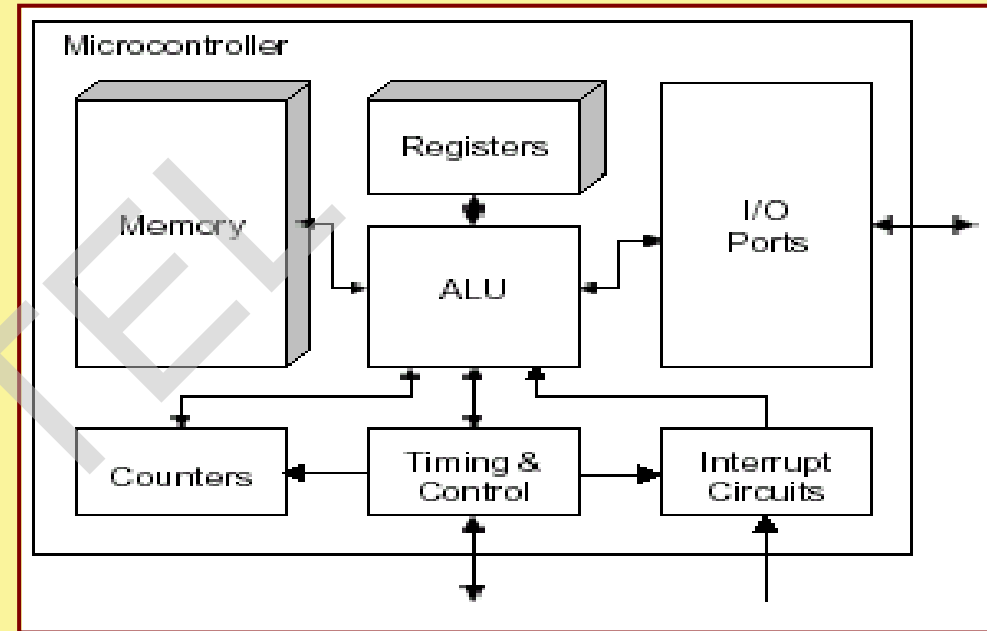
What is a Microcomputer?

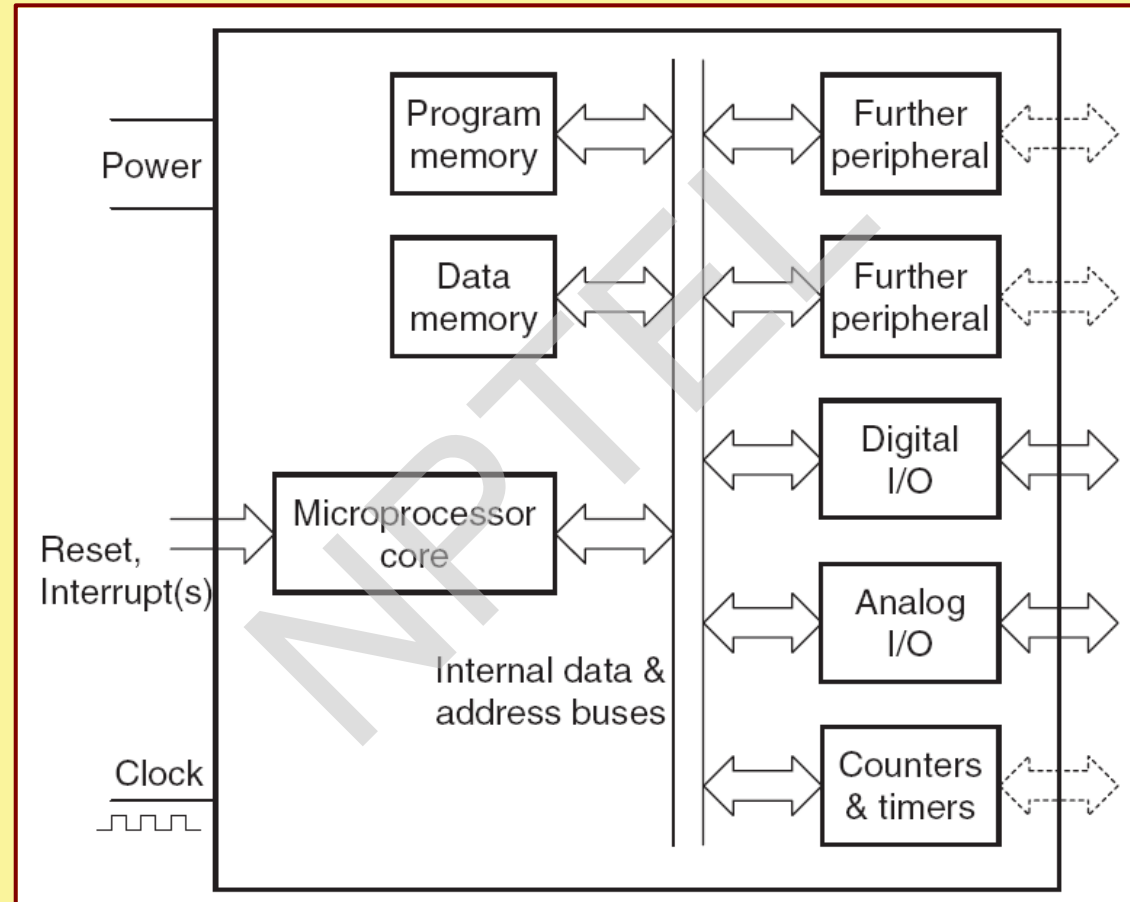
- It is a computer system built using a microprocessor.
- Since a microprocessor does not contain memory and I/O, we have to interface these to build a microcomputer.
 - Too complex and expensive for very small and low-cost embedded systems.



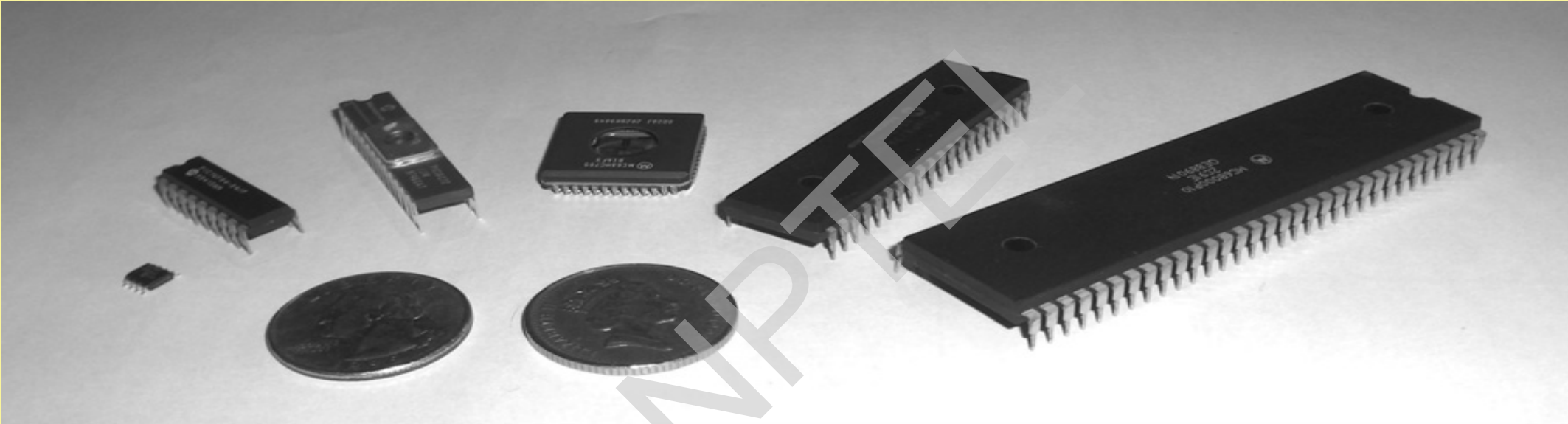
Microcontrollers: The Heart of Embedded Systems

- It is basically a computer on a single chip.
 - Very inexpensive, small, low power.
 - Convenient for use in embedded system design.
- It operates on data that are fed through its serial or parallel input ports, controlled by the software stored in on-chip memory.
 - Often has analog input pins, timers and other utility circuitry built-in.





Microcontroller Packaging and Appearance



From left to right:

PIC 12F508, PIC 16F84A, PIC 16C72, Motorola 68HC05B16, PIC 16F877, Motorola 68000

How Microcontrollers are different from PCs?

- When a PC executes a program, the program is first loaded from disk/SSD into an allocated section of memory.
 - Usually the program is loaded part by part to conserve memory space.
 - There is a complicated operating system that handles all low-level operations (includes low-level driver codes for interfacing with various devices).
- In a microcontroller there is no disk to read from.
 - On-chip ROM stores the program that is to be executed.
 - Size of the ROM limits the maximum size of the application.
 - There is no operating system, and the program in ROM is the only program that is running (must include low-level routines).

Where are Microcontrollers Used?

- Typically in applications where processing power is not critical.
 - Modern-day household can have 10 to 50 such devices embedded in various devices and equipments.
- One-third of the applications are in the office automation segment.
- Another one-third are in consumer electronics goods.
- Rest one-third are used in automotive and communication applications.

Evolution of Microcontrollers

- Microcontroller evolved from a microprocessor-based board-level design to a single chip in the mid-1970's.
 - As the process of miniaturization continued, all of the components needed for a controller were built into a single chip.
- In the mid-1980's, microcontrollers got embedded into a larger ASIC (Application Specific Integrated Circuit).
 - Microcontrollers are fabricated as a module inside a larger chip.

Advantages of using microcontrollers

- Fast and effective
 - The architecture correlates closely with the problem being solved (control systems).
- Low cost / Low power
 - High level of system integration within one component.
 - Only a handful of components needed to create a working system.
- Compatibility
 - Opcodes and binaries are the SAME for all 80x51 / ARM / PIC variants.



NPTEL ONLINE CERTIFICATION COURSES

**Thank
you!**



NPTEL ONLINE CERTIFICATION COURSES

Course Name: Embedded System Design with ARM

Faculty Name: Prof. Indranil Sen Gupta

Department : Computer Science and Engineering

Topic

Lecture 4: Architecture of ARM Microcontroller (Part 1)

CONCEPTS COVERED

- ☐ ARM series of microcontrollers
- ☐ Evolution of ARM microcontrollers
- ☐ Important ARM features

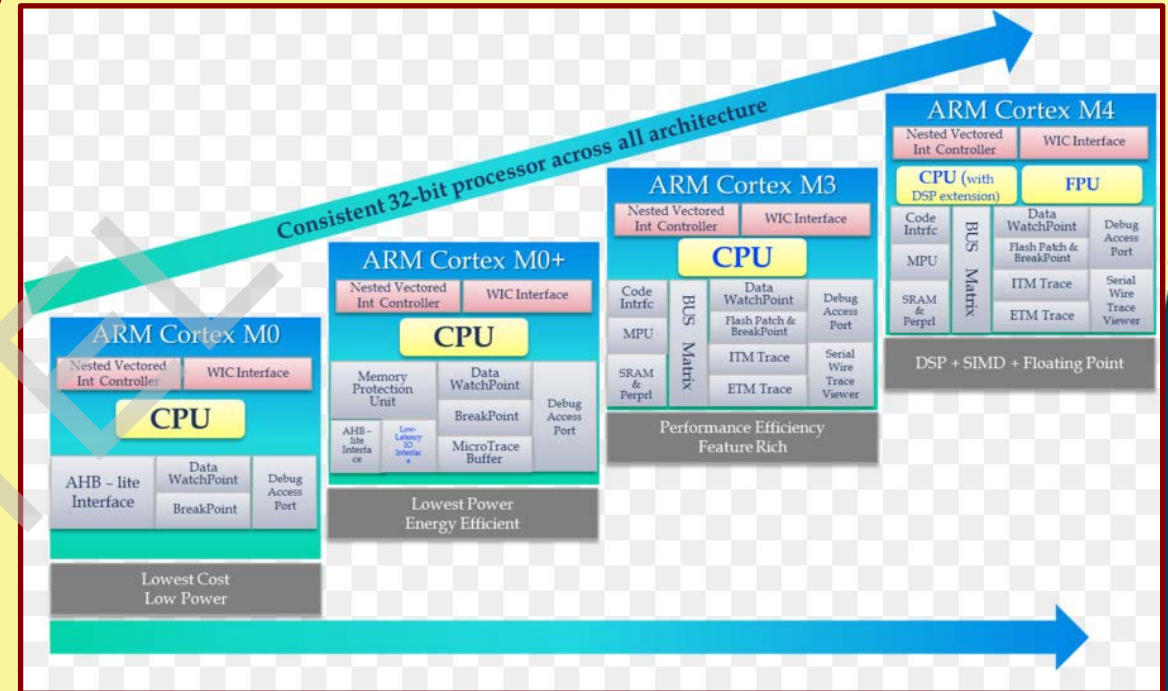


History of ARM Series of Microcontrollers

- Architectural ideas developed in 1983 by Acorn Computers.
 - To replace the 8-bit 6502 microprocessor in BBC computers.
 - The first commercial RISC implementation.
- The company founded in 1990.
 - Advanced RISC Machine (ARM).
 - Initially owned by Acorn, Apple and VLSI.

Why do we talk about ARM?

- One of the most widely used processor cores.
- Some application examples:
 - ARM7: iPod
 - ARM9: BenQ, Sony Ericsson
 - ARM11: Apple iPhone, Nokia N93, N100
 - 90% of 32-bit embedded RISC processors till 2010.
- Mainly used in battery-operated devices:
 - Due to low power consumption and reasonably good performance.



About ARM Processors

- A simple RISC-based architecture with powerful design.
- A whole family of ARM processors exist.
 - Share similar design principles and a common instruction set.
- Design philosophy:
 - Small processor for lower power consumption (for embedded system applications).
 - High code density for limited memory and physical size restrictions.
 - Can interface with slow and low-cost memory systems.
 - Reduced die size for processor to accommodate more peripherals.

Popular ARM Architectures

- ARM7
 - 3 pipeline stages (fetch/decode/execute)
 - High code density / low power consumption
 - Most widely used for low-end systems
- ARM9
 - Compatible with ARM7
 - 5 stages (fetch/decode/execute/memory/write)
 - Separate instruction and data cache
- ARM10
 - 6-stages (fetch/issue/decode/execute/memory/write)

ARM Family Comparison

	ARM 7 (1995)	ARM9 (1997)	ARM10 (1999)	ARM11 (2003)
Pipeline depth	3-stage	5-stage	6-stage	8-stage
Typical clock frequency (MHz)	80	150	260	335
Power (mW/MHz)	0.06	0.19	0.50	0.40
Throughput (MIPS/MHz)	0.97	1.1	1.3	1.2
Architecture	Non Neumann	Harvard	Harvard	Harvard
Multiplier	8 x 32	8 x 32	16 x 32	16 x 32

ARM is based on RISC Architecture

- RISC supports simple but powerful instructions that execute in a single cycle at high clock frequency.
- Major design features:
 - **Instructions**: reduced set / single cycle / fixed length
 - **Pipeline**: decode in one stage / no need for microcode
 - **Registers**: large number of general-purpose registers (GPRs)
 - **Load/Store Architecture**: data processing instructions work on registers only; load/store instructions to transfer data from/to memory.
- Now-a-days *CISC machines also implement RISC concepts.*

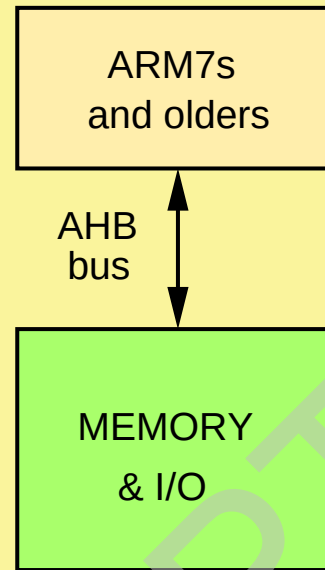
ARM Features

- ARM architecture is different from pure RISC:
 - Variable cycle execution for certain instructions (multiple-register load/store for higher code density).
 - In-line barrel shifter results in more complex instructions (improves performance and code density).
 - Thumb 16-bit instruction set (results in improvement in code density by about 30%).
 - Conditional execution (reduces branch and improves performance).
 - Enhanced instructions (some DSP instructions are present).

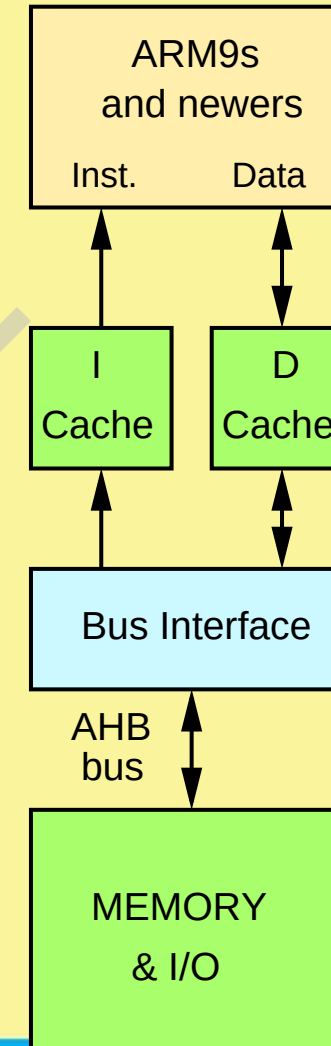
Memory-mapped I/O:

- No specific instructions for I/O
- Use Load/Store instr. for I/O
- Peripheral's registers at some memory addresses

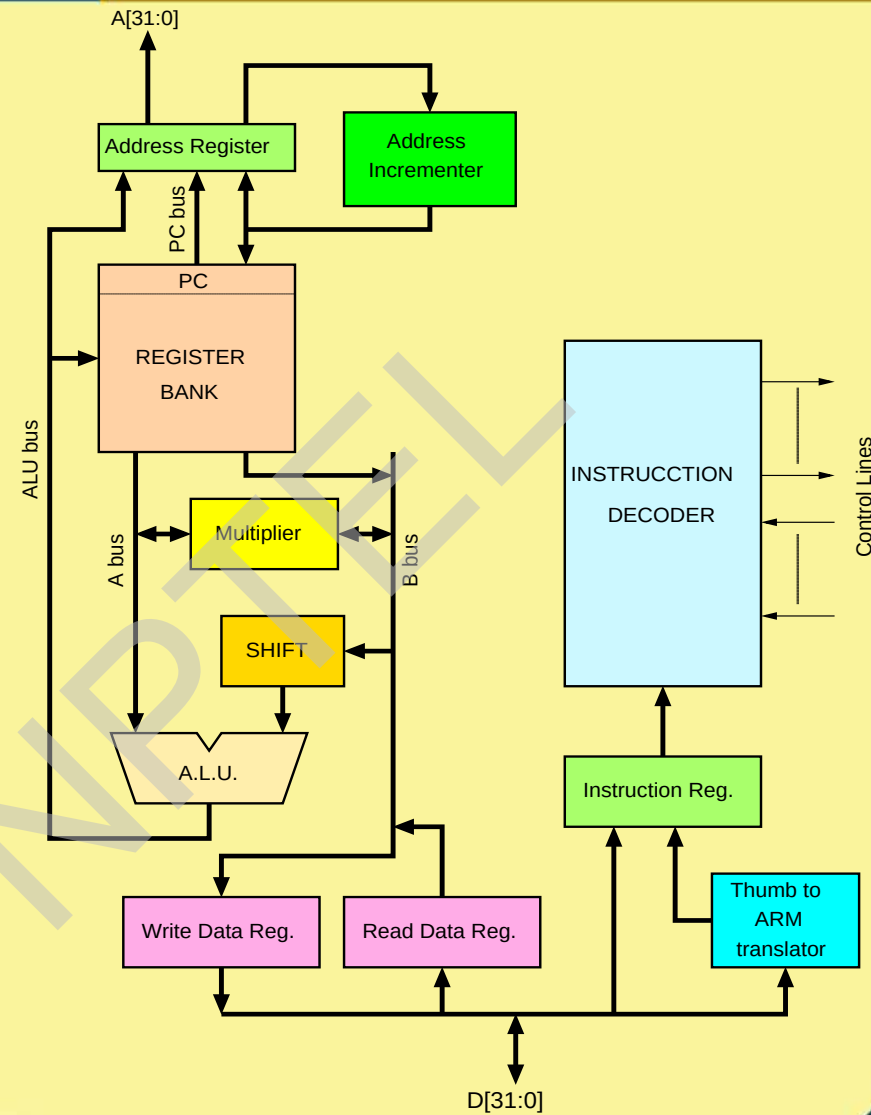
Von Neumann



Harvard



ARM7 Typical Architecture





NPTEL ONLINE CERTIFICATION COURSES

Thank
you!



NPTEL ONLINE CERTIFICATION COURSES

Course Name: Embedded System Design with ARM

Faculty Name: Prof. Indranil Sen Gupta

Department : Computer Science and Engineering

Topic

Lecture 5: Architecture of ARM Microcontroller (Part 2)

CONCEPTS COVERED

- ☐ Basic concepts of pipeline processing
- ☐ Pipeline speedup
- ☐ Instruction pipelining in ARM

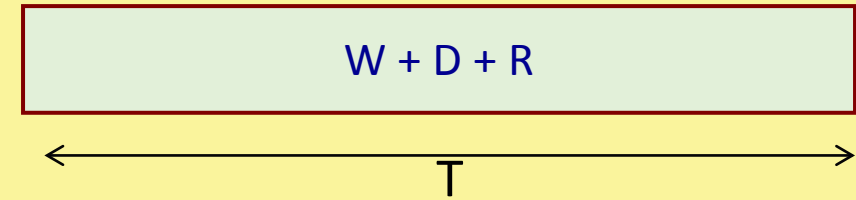


What is Pipelining?

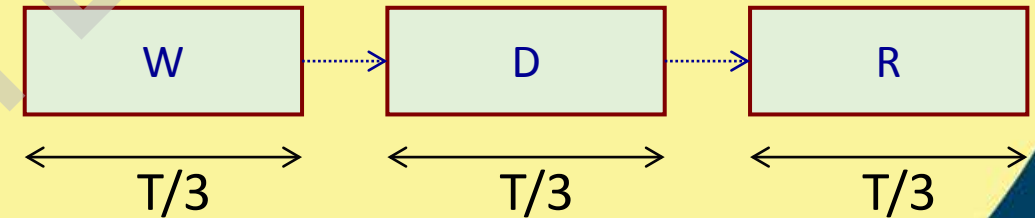
- A mechanism for overlapped execution of several input sets by partitioning some computation into a set of k sub-computations (or stages).
 - Very nominal increase in the cost of implementation.
 - Very significant speedup (ideally, k).
- Where are pipelining used in a computer system?
 - **Instruction execution**: Several instructions executed in some sequence.
 - **Arithmetic computation**: Same operation carried out on several data sets.
 - **Memory access**: Several memory accesses to consecutive locations are made.

A Real-life Example

- Suppose you have built a machine M that can wash (W), dry (D), and iron (R) clothes, one cloth at a time.
 - Total time required is T .
- As an alternative, we split the machine into three smaller machines M_W , M_D and M_R , which can perform the specific task only.
 - Time required by each of the smaller machines is $T/3$ (say).

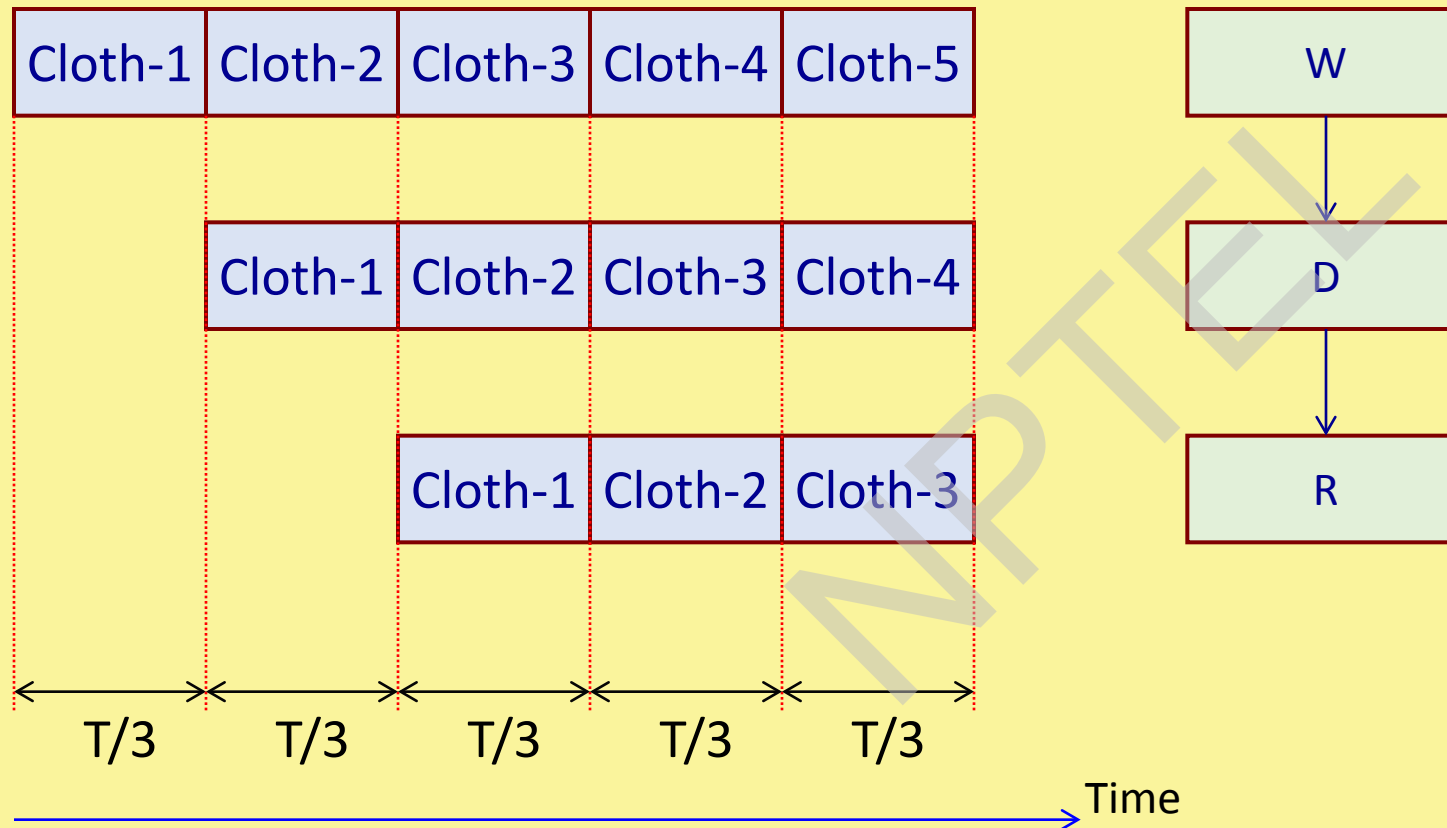


For N clothes, time $T_1 = N.T$



For N clothes, time $T_3 = (2 + N).T/3$

How does the pipeline work?



Finishing times:

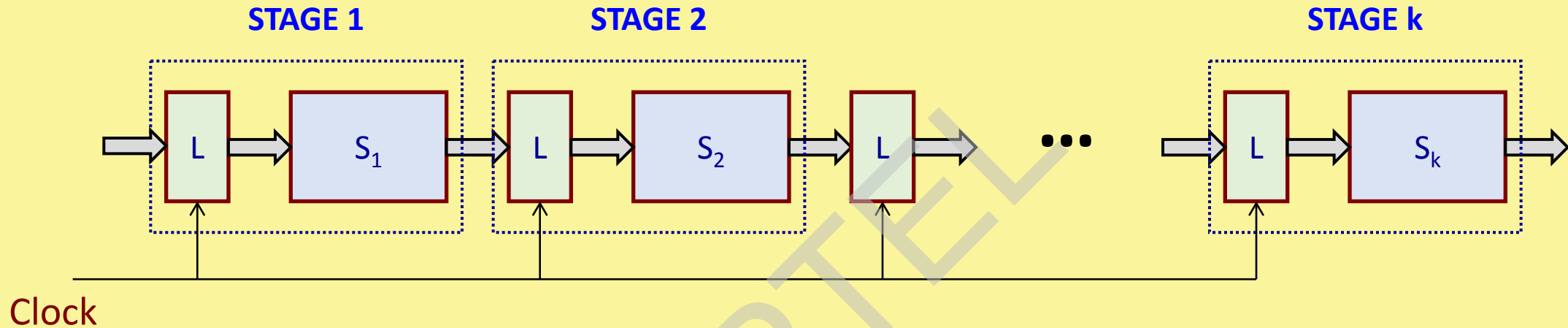
- Cloth-1 – $3.T/3$
- Cloth-2 – $4.T/3$
- Cloth-3 – $5.T/3$
- ...
- Cloth-N – $(2 + N).T/3$

Extending the Concept to Processor Pipeline

- The same concept can be extended to hardware pipelines.
- Suppose we want to attain k times speedup for some computation.
 - **Alternative 1:** Replicate the hardware k times $\rightarrow$ cost also goes up k times.
 - **Alternative 2:** Split the computation into k stages $\rightarrow$ very nominal cost increase.
- Need for buffering:
 - In the washing example, we need a tray between machines (W & D, and D & R) to keep the temporarily before it is accepted by the next machine.
 - Similarly in hardware pipeline, we need a *latch* between successive stages to hold the intermediate results temporarily.

cloth

Model of a Synchronous k-stage Pipeline



- The latches are made with master-slave flip-flops, and serve the purpose of isolating inputs from outputs.
- The pipeline stages are typically combinational circuits.
- When *Clock* is applied, all latches transfer data to the next stage simultaneously.

Speedup and Efficiency

Some notations:

τ :: clock period of the pipeline

t_i :: time delay of the circuitry in stage S_i

d_L :: delay of a latch

Maximum stage delay

$$\tau_m = \max \{t_i\}$$

Thus,

$$\tau = \tau_m + d_L$$

Pipeline frequency

$$f = 1 / \tau$$

- If one result is expected to come out of the pipeline every clock cycle, f will represent the maximum throughput of the pipeline.*

- The total time to process N data sets is given by

$$T_k = [(k - 1) + N].\tau$$

$(k - 1) \tau$ time required to fill the pipeline

1 result every τ time after that $\rightarrow$ total $N.\tau$

- For an equivalent non-pipelined processor (i.e. one stage), the total time is

$$T_1 = N.k.\tau$$

(ignoring the latch overheads)

- Speedup of the k -stage pipeline over equivalent non-pipelined processor:

$$S_k = \frac{T_1}{T_k} = \frac{N.k.\tau}{k.\tau + (N - 1).\tau} = \frac{N.k}{k + (N - 1)}$$

As $N \rightarrow \infty$, $S_k \rightarrow k$

- Pipeline efficiency:

- How close is the performance to its ideal value?

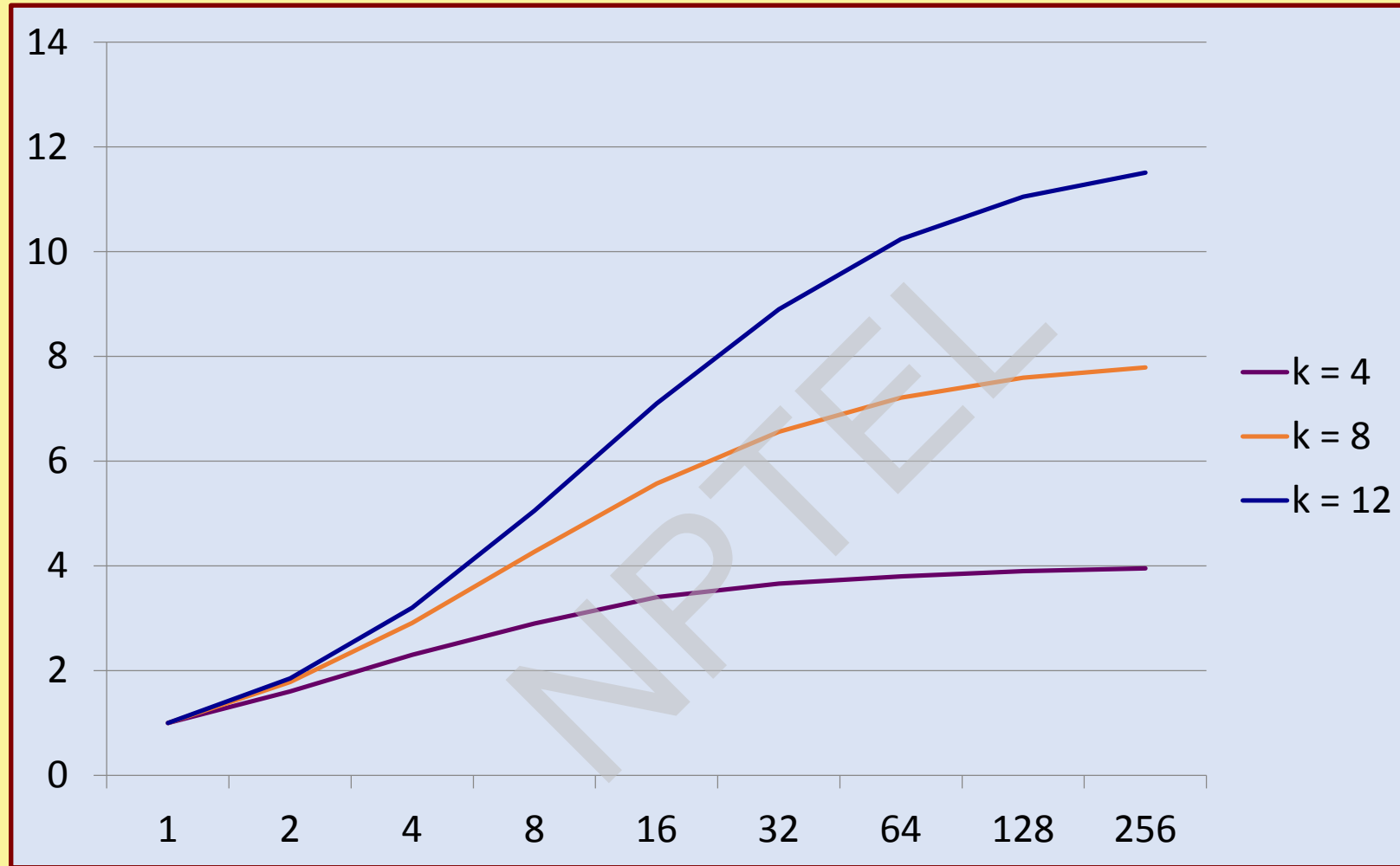
$$E_k = \frac{S_k}{k} = \frac{N}{k + (N - 1)}$$

- Pipeline throughput:

- Number of operations completed per unit time.

$$H_k = \frac{N}{T_k} = \frac{N}{[k + (N - 1)].\tau}$$

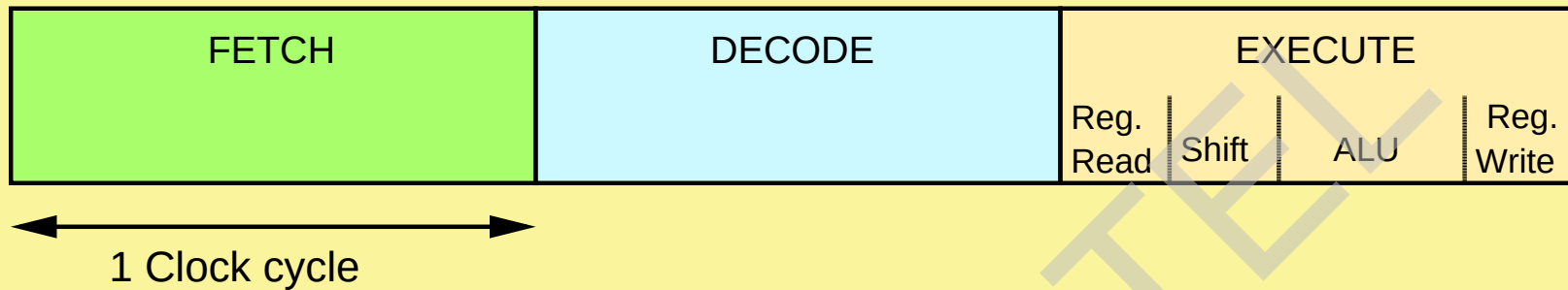
Speedup



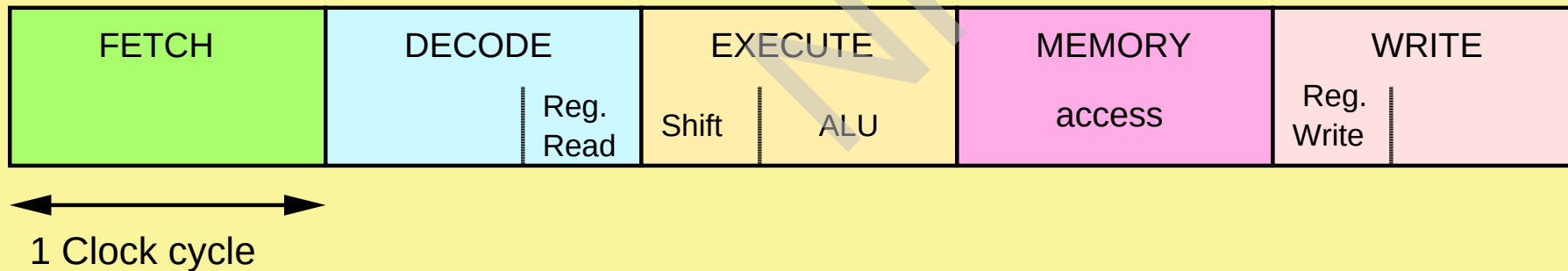
Number of tasks N

ARM Pipelining Examples

ARM7TDMI Pipeline

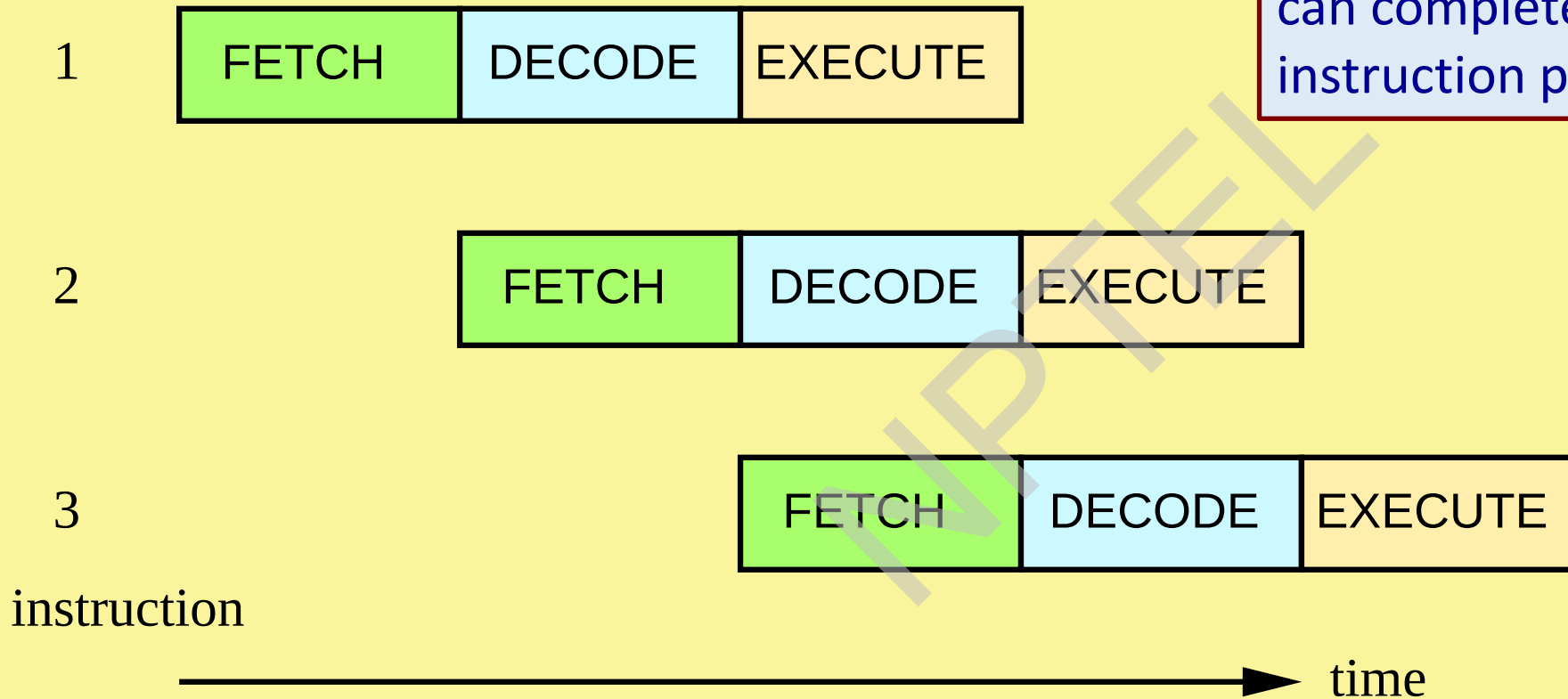


ARM9TDMI Pipeline



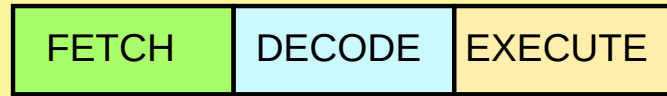
Pipelining in ARM7

Simple instructions (like ADD, SUB) can complete at a rate of one instruction per cycle.

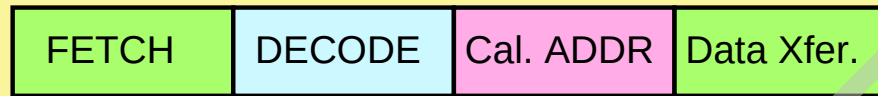


With more complex instructions ... stall cycles possible

1 ADD



2 STR



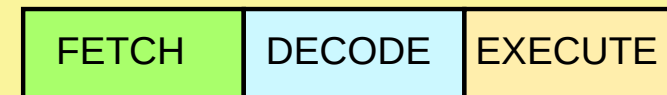
3 ADD



4 ADD



5 ADD



instruction

time

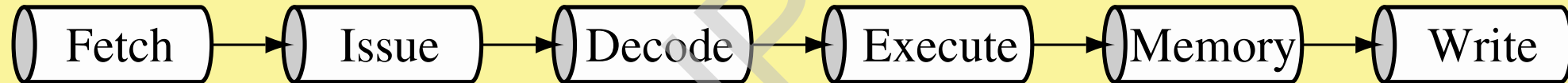
time



ARM7 3-state Pipeline

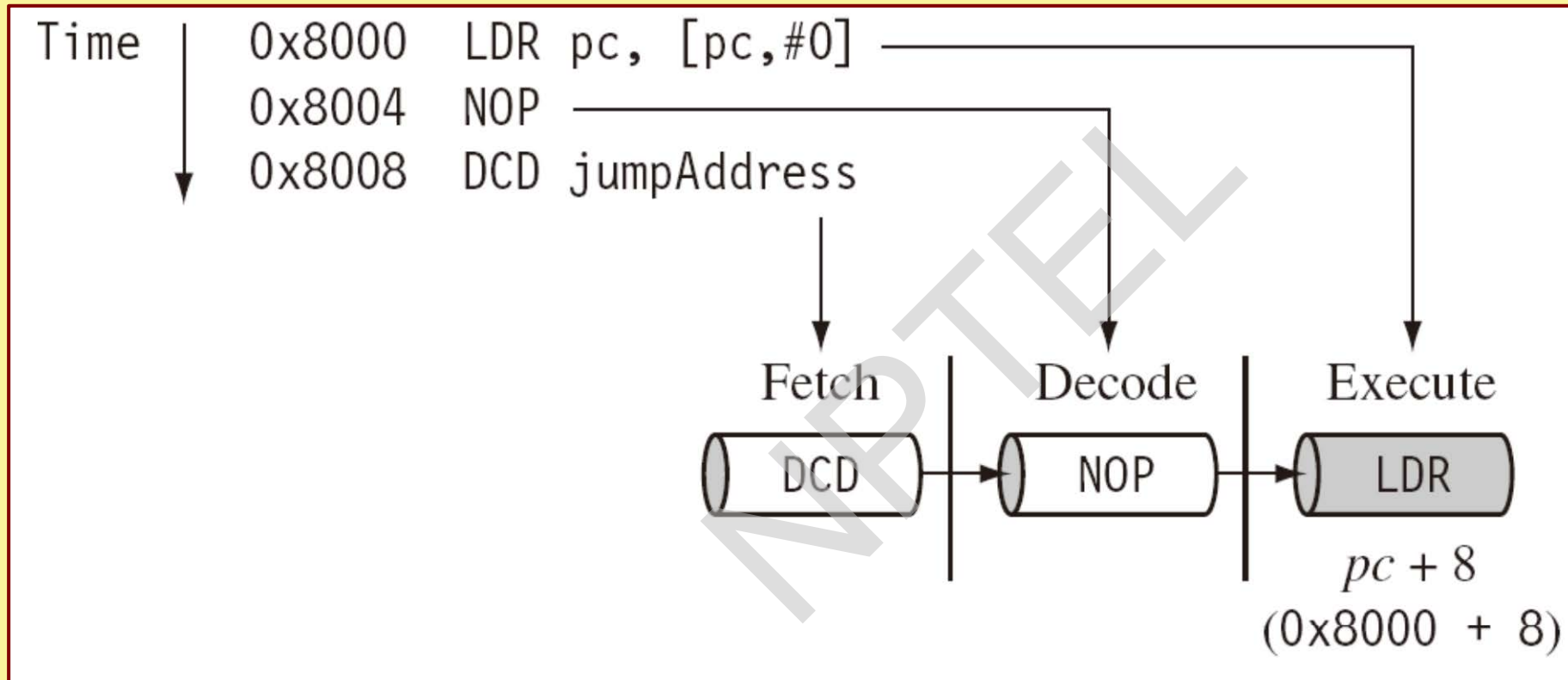


ARM9 5-state Pipeline



ARM10 6-state Pipeline

- In execution, the program counter (PC) is always 8 bytes ahead.





NPTEL ONLINE CERTIFICATION COURSES

Thank
you!



NPTEL ONLINE CERTIFICATION COURSES

Course Name: Embedded System Design with ARM

Faculty Name: Prof. Indranil Sen Gupta

Department : Computer Science and Engineering

Topic

Lecture 6: Architecture of ARM Microcontroller (Part 3)

CONCEPTS COVERED

- ☐ ARM processor modes and registers
- ☐ Special registers and exception handling
- ☐ ARM and Thumb modes of execution



Processor Modes

Processor Mode	Code	Description
User	usr	Normal program execution mode
FIQ	fiq	Entered when a high priority (fast) interrupt is raised
IRQ	irq	Entered when a low-priority (normal) interrupt is raised
Supervisor	svc	A protected mode for the operating system (entered on reset and when software interrupt instruction is executed)
Abort	abt	Used to handle memory access violations
Undefined	und	Used to handle undefined instructions
System	sys	Runs privileged operating system tasks

Registers

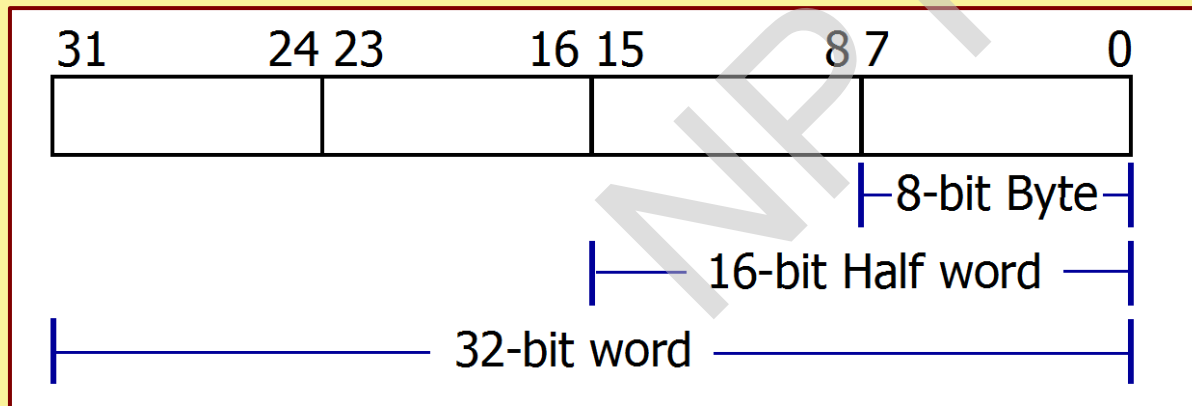
- ARM has 37 registers all of which are 32-bits long.
- These registers are:
 - 1 dedicated *program counter* (PC)
 - 1 dedicated *current program status register* (CPSR)
 - 5 dedicated *saved program status registers* (SPSR)
 - 30 *general-purpose registers* (GPR)

Registers (contd.)

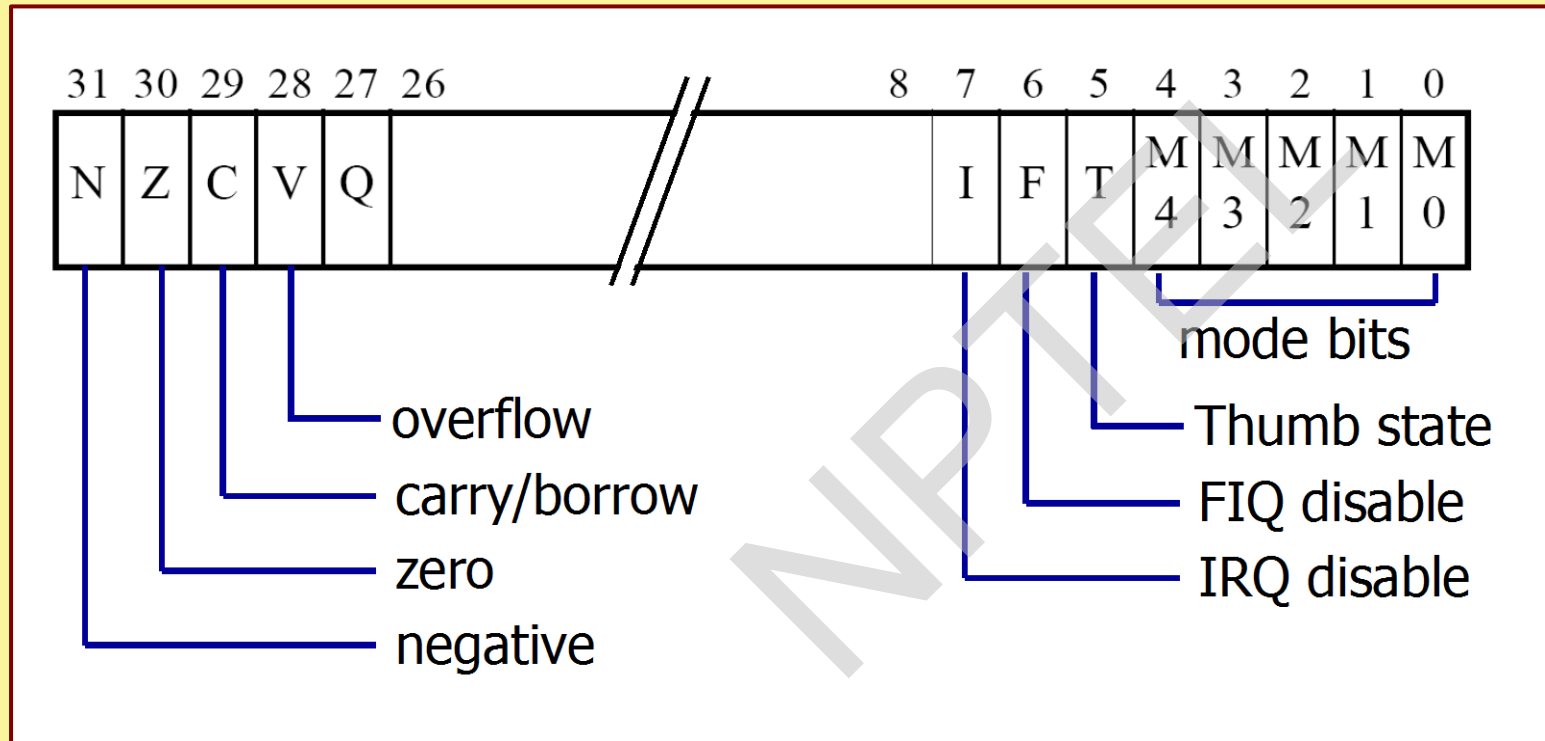
- The current processor mode governs which of several register sets is accessible.
- Only 16 registers are visible to a specific mode of operation. Each mode can access:
 - A particular set of registers r0-r12
 - r13 (SP, stack pointer)
 - r14 (LR, link register)
 - r15 (PC, program counter)
 - Current program status register (CPSR)
- Privileged modes (except System) can also access a particular SPSR.

General-purpose Registers

- 6 data types are supported (signed/unsigned)
 - 8-bit byte, 16-bit half-word, 32-bit word
- All ARM operations are 32-bit.
 - Shorter data types are only supported by data transfer operations.



Current Program Status Register (CPSR)



Special Registers

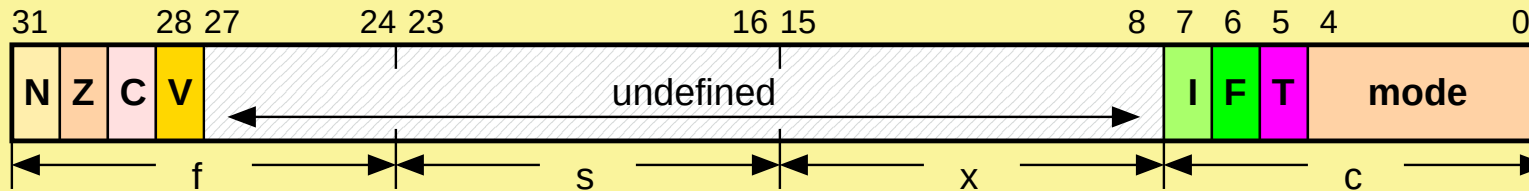
- **PC (r15)**: Any instruction with PC as its destination register is a program branch.
- **LR (r14)**: Saves a copy of PC when executing the *BL instruction* (subroutine call) or when jumping to an exception or an interrupt handler.
 - It is copied back to PC on return from those routines.
- **SP (r13)**: There is no stack in the ARM architecture.
 - R13 is reserved as a pointer for the software-managed stack.
- **CPSR**: Holds the visible status register.
- **SPSR**: Holds a copy of the previous status register while executing exception or interrupt handler routines.
 - Copied back to CPSR on return from exception or interrupt.

Program Counter

- When the processor is executing in *ARM mode*:
 - All instructions are 32-bits wide, and must be word aligned.
 - The last two bits of PC are zero (i.e. not used).
 - Due to pipelining, PC points 8 bytes ahead of the current instruction, or 12 bytes ahead if the current instruction includes a register-specified shift.
- When the processor is executing in *Thumb mode*:
 - All instructions are 16-bits wide, and are half-word aligned.
 - The last bit of the PC is zero (i.e. not used).

Register Organization Summary

User, SYS	FIQ	IRQ	SVC	Undef	Abort
r0	User mode r0-r7, r15, and cpsr	User mode r0-r12, r15, and cpsr	User mode r0-r12, r15, and cpsr	User mode r0-r12, r15, and cpsr	User mode r0-r12, r15, and cpsr
r1					
r2					
r3					
r4					
r5					
r6					
r7					
r8	r8	r13 (sp) r14 (lr)	r13 (sp) r14 (lr)	r13 (sp) r14 (lr)	r13 (sp) r14 (lr)
r9	r9				
r10	r10				
r11	r11				
r12	r12				
r13 (sp)	r13 (sp)				
r14 (lr)	r14 (lr)				
r15 (pc)					
cpsr					
	spsr	spsr	spsr	spsr	spsr



Program Status Register

Condition code flags

- N = **N**egative result from ALU
- Z = **Z**ero result from ALU
- C = ALU operation **C**arried out
- V = ALU operation o**V**erflowed

Interrupt Disable bits.

- I = 1: Disables the IRQ.
- F = 1: Disables the FIQ.

T Bit (Arch. with Thumb mode only)

- T = 0: Processor in ARM state
- T = 1: Processor in Thumb state

Mode bits

10000	User
10001	FIQ
10010	IRQ
10011	Supervisor
10111	Abort
11011	Undefined
11111	System

Exception Handling

- When an exception occurs, the processor
 - Copies CPSR into SPSR_<mode>
 - Sets appropriate bits in CPSR
 - Changes to ARM state
 - Changes to related mode
 - Disables IRQ, FIQ
 - Stores return address in LR_<mode>
 - Sets PC to vector address
- To return, the exception handler needs to:
 - Restore CPSR from SPSR_<mode>
 - Restore PC from LR_<mode>

	• • • •
0x1C	FIQ
0x18	IRQ
0x14	(Reserved)
0x10	Data Abort
0x0C	Prefetch Abort
0x08	Software Interrupt
0x04	Undefined Instruction
0x00	Reset

Vector Table

- Exception handling in ARM is controlled through an area of memory called the *vector table*.
 - Exists at the bottom of the memory map *from 0x00 to 0x1c*.
 - Within this table, one word is allocated to each of the various exception types.
 - This word will contain some form of ARM instruction that should perform a branch.
 - Does not contain an address.

ARM and Thumb Instruction Set

- Most ARM implementations provide two instruction sets:
 - a) 32-bit ARM instruction set
 - b) 16-bit Thumb instruction set

	ARM (cpsr T = 0)	Thumb (cpsr T = 1)
Instruction size	32-bit	16-bit
Core instructions	58	30
Conditional execution	Most	Only branch instructions
Data processing instructions	Access to barrel shifter and ALU	Separate barrel shifter and ALU instructions
Program status register	Read-write in privileged mode	No direct access
Register usage	15 GPRs + pc	8 GPRs + 7 high registers + pc

What is Conditional Execution?

- It controls whether or not the CPU will execute an instruction.
- Most instructions have a condition attribute that determines whether it will be executed based on the status of the condition flags.
 - Prior to execution, the processor compares the condition attribute with the condition flags in CPSR.
 - If they do not match, the instruction is not executed.
 - Example:

MOVEQ r1,#0 (if zero flag is set, then r1 = 0)

- The condition attribute (here *EQ*) is suffixed to the instruction mnemonic.



NPTEL ONLINE CERTIFICATION COURSES

Thank
you!