Embedded Hardware: Hardware Building Blocks

Basics

Topics to be discussed

- Basic Notations describing hardware design
  - Diagrams
  - Symbols
- Embedded Board
- Von-Neumann Model
- Semiconductors
- Active Building Blocks of Processors and Memory

Engineering Hardware drawings

Created by hardware engineers

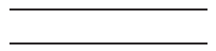
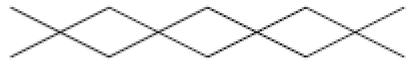
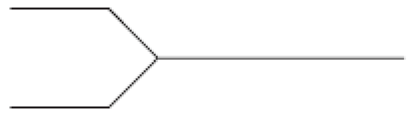
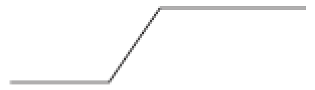
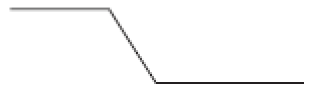
Used to describe the hardware designs to the outside world.

- **Block diagram**
  - Major components of board – processors, buses, I/O, memory
  - A basic overview of the hardware, with implementation details abstracted out.
  - **Pros:** Simple, basic overview – basis for detailed hardware diagrams
  - **Cons:** Not detailed enough for software engineer
- **Schematics**
  - Electronic circuit diagrams that provide a more detailed view of all of the devices within a circuit or within a single component—everything from processors down to resistors.
  - Depicts the flow of data in the system.
  - Schematic symbols are used to depict all components.
  - Helpful in debugging – hardware and software.

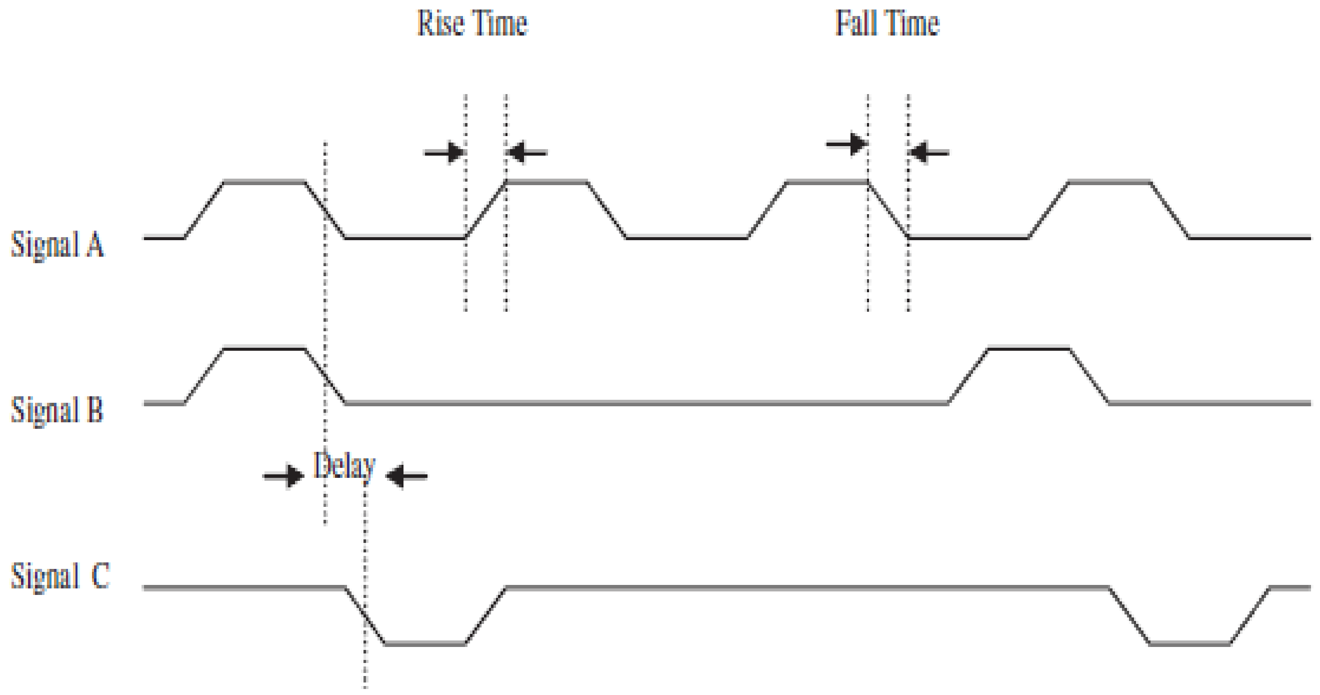
- ***Wiring diagrams***
  - Represent the *bus (represented by* vertical and horizontal lines*)* connections between the major and minor components on a board or within a chip.
  - Approximate depiction of the physical layout of component/board.
- **Logic diagrams**
  - Show a wide variety of circuit information using logical symbols (AND, OR, NOT, XOR, and so on), and logical inputs and outputs (the 1’s and 0’s).
- **Timing diagrams**
  - display timing graphs of various input and output signals of a circuit, as well as the relationships between the various signals.

Symbols, Conventions & Rules

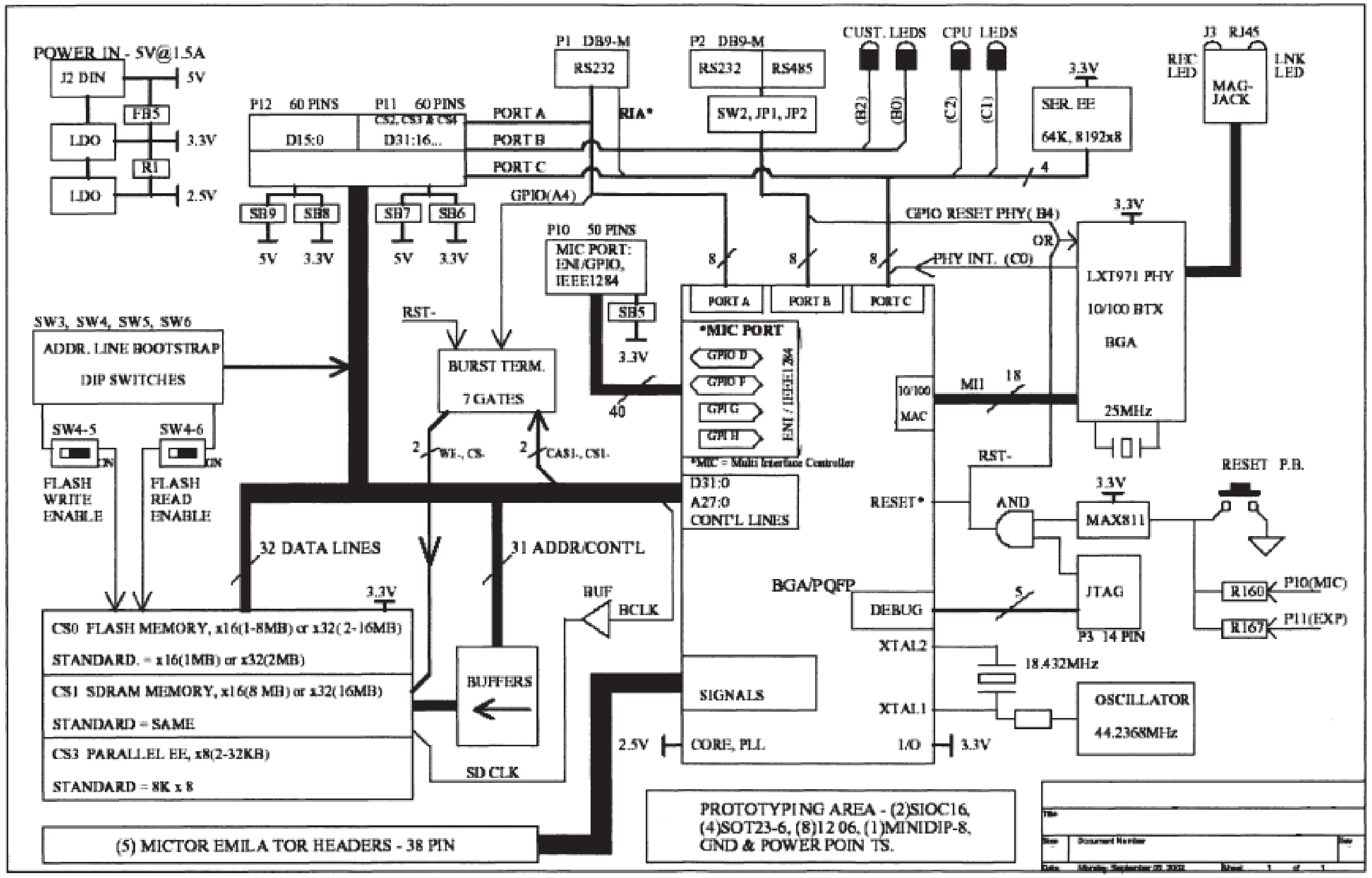
- Regardless of the type, in order to understand how to read and interpret these diagrams, it is first important to *learn* the standard **symbols**, **conventions**, and **rules** used.
- *Timing diagrams symbol table*

Symbol	Input Signals	Output Signals
	Input signal must be valid	Output signal will be valid
	Input signal doesn't affect sys-tem, will work regardless	Indeterminate output signal
	Garbage signal (nonsense)	Output signal not driven (floating), tristate, HiZ, high impedance
	If the input signal rises	Output signal will rise
	If the input signal falls	Output signal will fall

Timing diagram example



Schematic diagram example



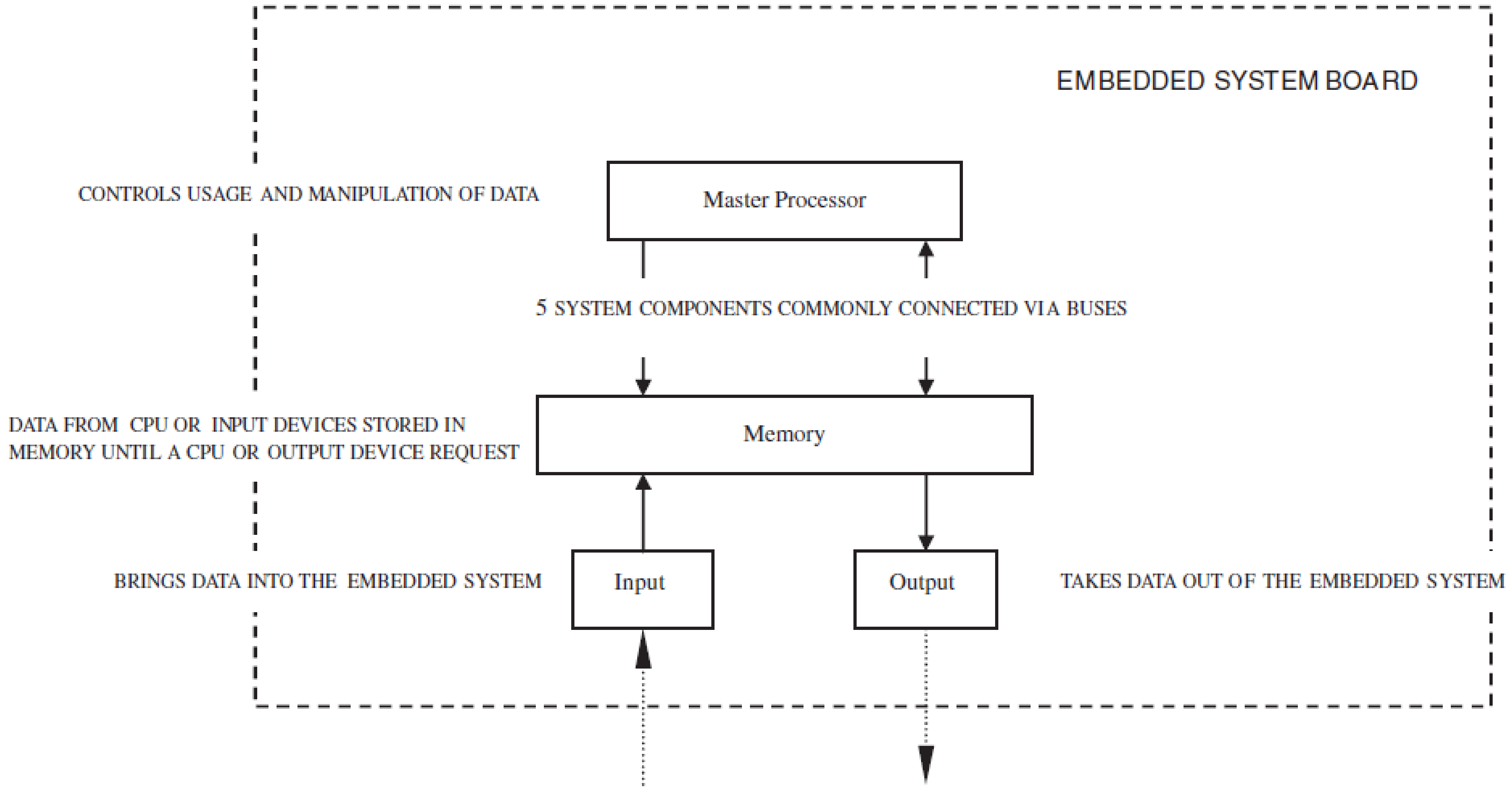
Schematic Diagram – Conventions:

- Title section : Name, hardware engineer, date, list of revisions
- Schematic symbols
- Labels – pin numbers, IC, size, type, power etc.
- Abbreviatons and prefixes: Eg. k –kilo, M – Mega
- Functional groups of components
- I/O and Voltage source/Ground terminals

The Embedded Board and the von Neumann Model

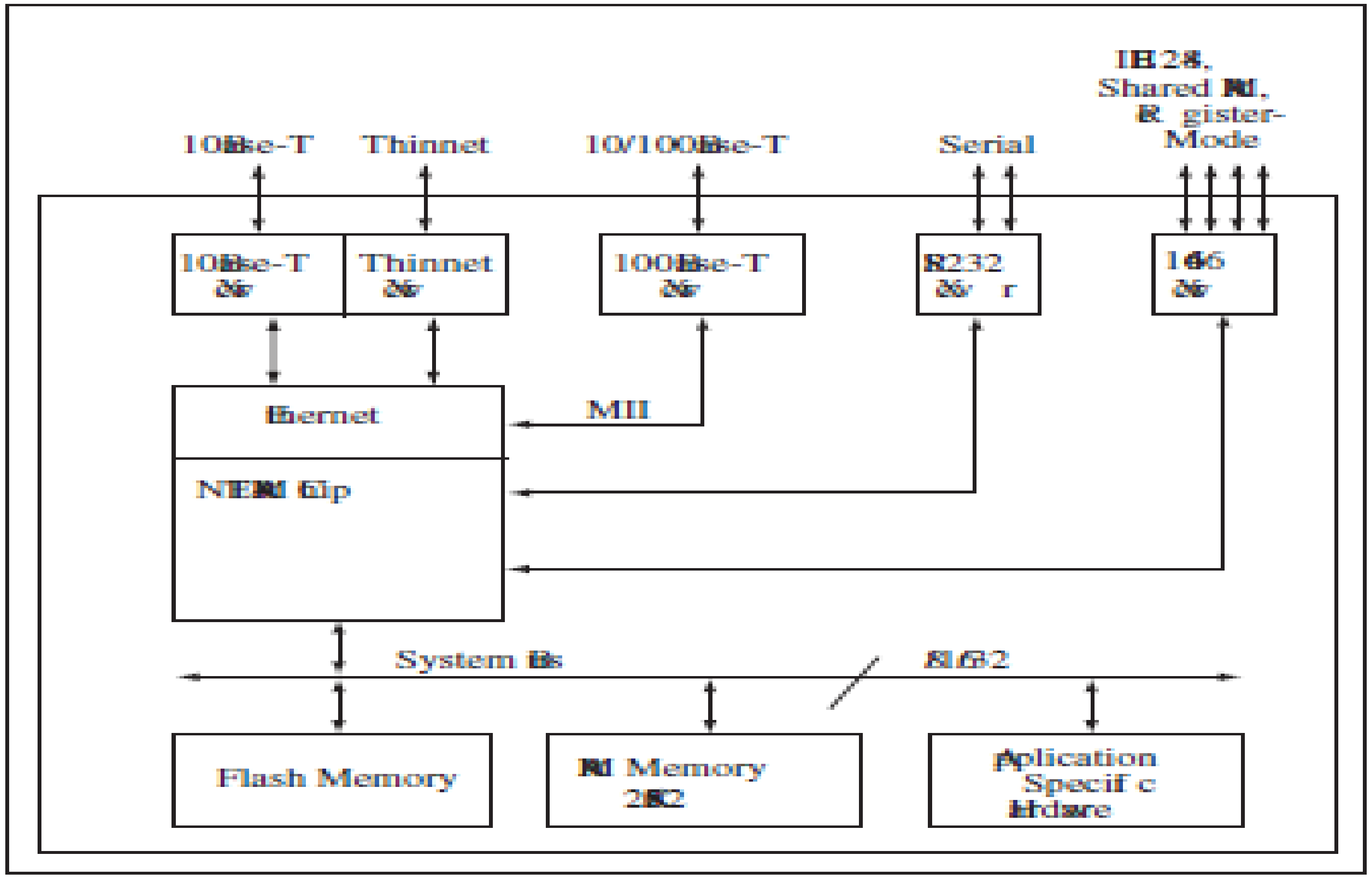
- In embedded devices, all the electronics hardware resides on a board, called printed circuit board (PCB).
- Major hardware components of most boards can be classified into five major categories:
  - **Central Processing Unit (CPU)** – the master processor
  - **Memory** – where the system’s software is stored
  - **Input Device(s)** – input slave processors and relative electrical components
  - **Output Device(s)** – output slave processors and relative electrical components
  - **Data Pathway(s)/Bus(es)** – interconnects the other components, providing a “highway” for data to travel on from one component to another, including any wires, bus bridges, and/or bus controllers

Von Neumann model


EMBEDDED SYSTEM BOARD
CONTROLS USAGE AND MANIPULATION OF DATA
Master Processor
5 SYSTEM COMPONENTS COMMONLY CONNECTED VIA BUSES
DATA FROM CPU OR INPUT DEVICES STORED IN MEMORY UNTIL A CPU OR OUTPUT DEVICE REQUEST
Memory
BRINGS DATA INTO THE EMBEDDED SYSTEM
Input
Output
TAKES DATA OUT OF THE EMBEDDED SYSTEM

Embedded system board organization

Net Silicon ARM7 reference board



Basic Components

- Before major components of embedded board, lets see basic components.
- Embedded board is made of interconnected basic electronic devices
  - Wires, resistors, capacitors, inductors, and diodes
- Components can be classified into
  - **Active components:** components capable of delivering, receiving and storing power.

    Eg: transistors, diodes
  - **Passive components:** components which can only receive and store power

    Eg.: wires, capacitors, resistors

Basic Hardware Materials

- **Conductors:** fewer impediments to an electric current. Eg. Metals – easily loose/gain valence electrons
- **Insulators:** impede electric current. Paper – less likely to loose/gain valence electrons
- **Semiconductors:** base elements have a conductive nature that can be altered by introducing other elements into their structure

Semiconductors

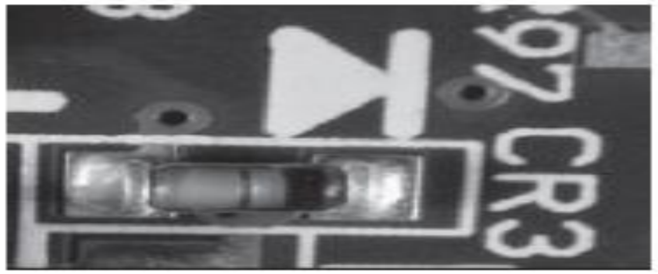
- Materials whose base elements have a conductive nature that can be altered by introducing other elements (impurities) into their structure.
- Example- silicon, germanium etc.
- It can be of 2 basic types:
  - P-type
    - Impurities called acceptors, such as boron, produce a shortage of electrons, creating a P-type semiconductor material.
  - N-type
    - Impurities (like arsenic, phosphorus, antimony, etc.), called donors, create a surplus of electrons creating an N-type semiconductor.

Active Building Blocks of Processors and Memory

- P-type and N-type semiconductors must be combined to be able to do anything practical.
- When P-type and N-type semiconductors are combined, the contact point, called the P-N Junction,
  - Acts as a one-way gate
  - Allow electrons to flow within the device in a direction dependent on the polarity of the materials.
- P and N-type form some of the most common basic electronic devices that act as the main building blocks in processor and memory chips:
  - Diodes
  - Transistors

(Cntd...)

- **Diodes-**
  - Semiconductor device made up of two materials, one P-type and one N-type joined together. A terminal is connected to each of the materials, called an anode and a cathode.



Diode and light emitting diode (LED)

- Forward biasing
  - Current flows through a diode from the anode to cathode as long as the anode has a higher (positive) voltage
- Reverse biasing
  - When current will not flow through the diode because the cathode has a higher (positive) voltage than the anode

(Cntd…)

- **Transistor**
  - current-**tran**sferring re**sistor**
  - Made up of some combination of P-type and N-type semiconductor material, with three terminals(*emitter*, *base*, and a *collector*) connecting to each of the three materials.
  - used for a variety of purposes,
    - current amplifiers (amplification),
    - in oscillators (oscillation),
    - in high-speed integrated circuits (ICs)
    - and/or in switching circuits(DIP switches, push buttons)

(Cntd…)

- Two main types of transistors are:
  - Bipolar junction transistor (BJT)
    - Made up of three alternating types of P-type and N-type material,
    - Are sub-classed based on the combination of these materials.
      - <u>NPN BJT</u>
        - is made up of two sections of N-type material, separated by a thin section of P-type material
      - <u>PNP BJT</u>
        - is made up of two sections of P-type materials, separated by a thin section of N-type material

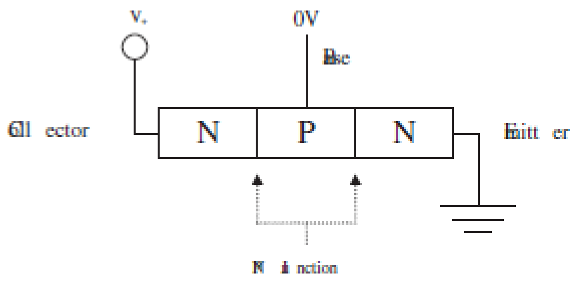


Figure 3-20a: NPN BJT "OFF"

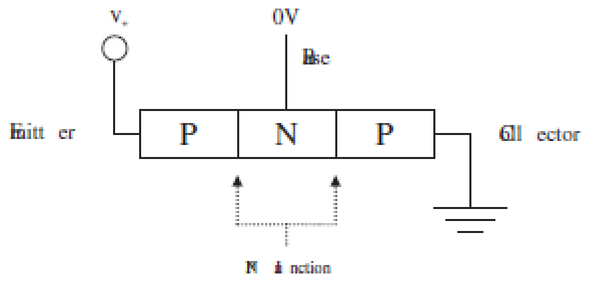


Figure 3-20b: PNP BJT "OFF"

(Cntd…)

- Field effect transistor (FET)
  - Made up of some combination of P-type and N-type semiconductor material, with three terminals
  - The terminals are called a source, a drain/sink, and a gate.
  - FETs do not require a biasing current, and are controlled via voltage alone.
  - The 2 most common types:
    - Metal-Oxide-Semiconductor FET(MOSFET)
    - Junction field-effect transistor (JFET).

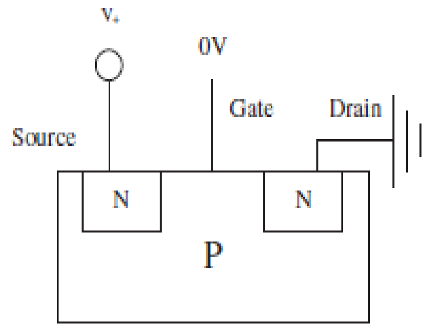


Figure 3-23a:
N-channel enhancement MOSFET "OFF"

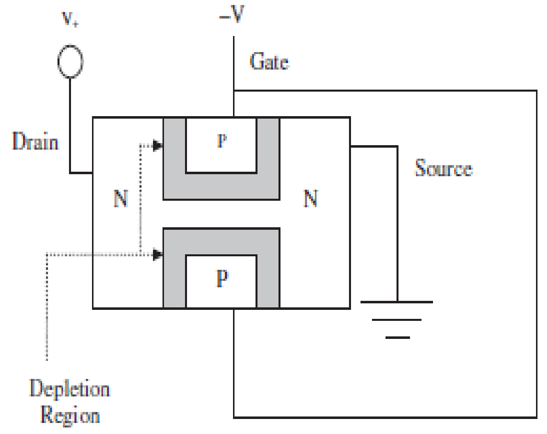


Figure 3-23b:
N-channel depletion MOSFET "OFF"

Embedded Processors

Topics to be discussed:

- Introduction
- ISA Architecture Models
- Internal processor design
- Processor Performance

Embedded Processors- Introduction

- Processors:
  - main functional units
  - primarily responsible for processing instructions and data.
- An electronic device contains at least one master processor,
- There can be additional slave processors, controlled by the master processor.
- Slave processors may either extend the instruction set of the master processor or act to manage memory, buses, and I/O devices.

- The complexity of the master processor usually determines whether it is classified as a ***microprocessor*** or a ***microcontroller***.
- Traditionally, Microprocessors contain a minimal set of integrated memory and I/O components
- Microcontrollers have most of the system memory and I/O components integrated on the chip.
- Though traditional definitions may not strictly apply to recent processor designs.

Architectures

- Embedded processors can be separated into various “groups” called architectures.
- What differentiates one processor group’s architecture from another is the **set of machine code instructions** that the processors within the architecture group can execute.
- Processors are considered to be of the same architecture when they can execute the same set of machine code instructions.

ISA Architecture Models

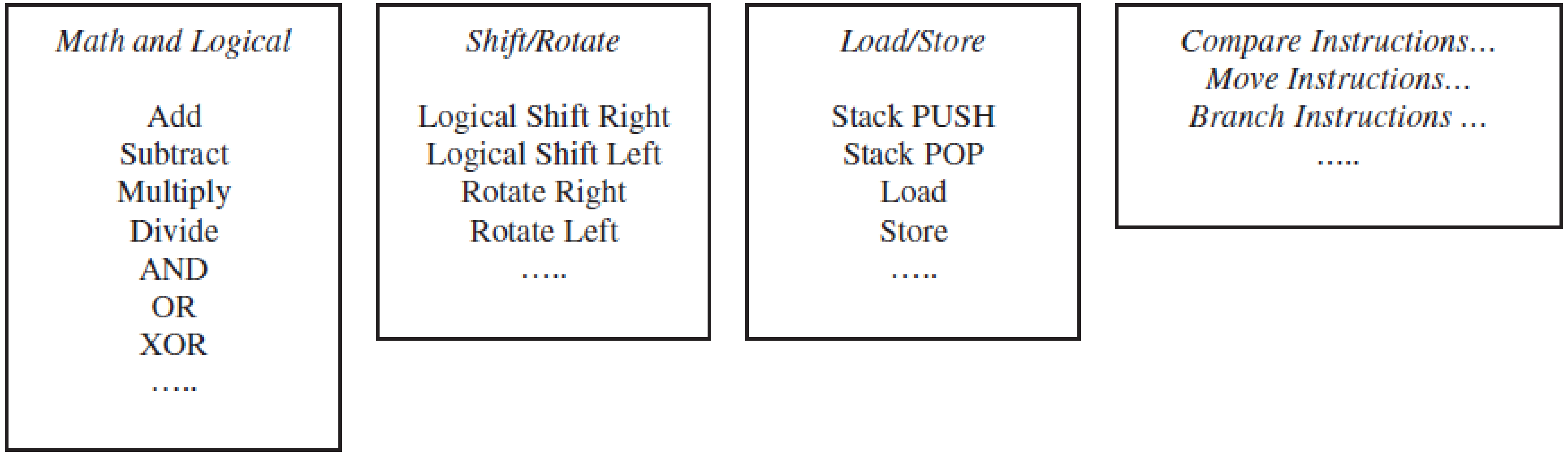
Instruction Set Architecture or ***ISA***

- The *features* that are built into an architecture's instruction set*.*
- defines such features as:
    - the operations,
    - the operands (data),
    - storage,
    - addressing modes, and
    - the handling of interrupts.

Operations

- *Operations* are made up of one or more instructions that execute certain commands.
- Different processors can execute **the exact same operations using a different number and different types of instructions**
- Functions that can be performed on the data, and they typically include computations (math operations), movement, branches, input/output operations, and context-switching operations
- Eg.: 8051 – over 100 instructions for all operations

Operations



Math and Logical

Add
Subtract
Multiply
Divide
AND
OR
XOR
…..

Shift/Rotate

Logical Shift Right
Logical Shift Left
Rotate Right
Rotate Left
…..

Load/Store

Stack PUSH
Stack POP
Load
Store
…..

Compare Instructions…
Move Instructions…
Branch Instructions …
…..

Figure 4-2a: Sample ISA operations

Operation Formats

- The format of an operation is the actual number and combination of bits (1’s and 0’s) that represent the operation.
- Referred as ‘Operation code’ or ‘opcode’

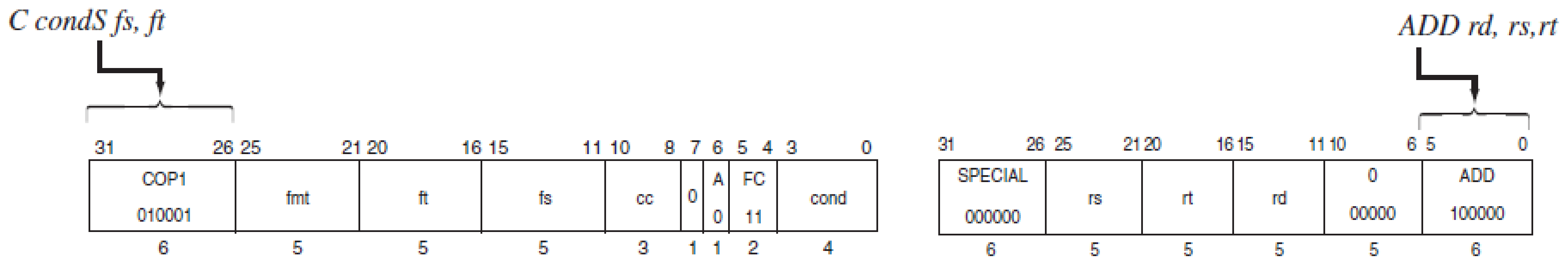


Figure 4-3b: MIPS32/MIPS I “CMP” and “ADD” operation sizes and locations [4-4]

Operands

- Data that operations manipulate.
- An ISA defines the types and formats of operands for a particular architecture.

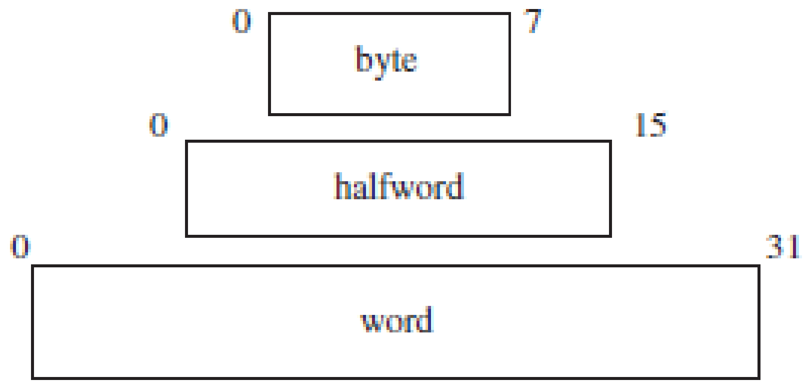
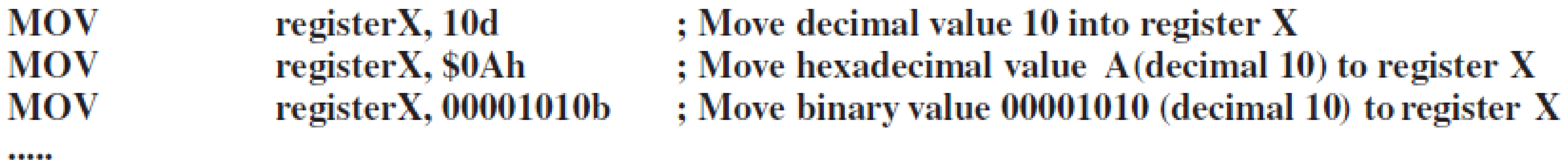


Figure 4-4: Simple operand types

Operand Format

- An ISA also defines the operand formats (how the data looks) that a particular architecture can support, such as binary, decimal and hexadecimal



```
MOV     registerX, 10d          ; Move decimal value 10 into register X
MOV     registerX, $0Ah         ; Move hexadecimal value  A(decimal 10) to register X
MOV     registerX, 00001010b    ; Move binary value 00001010 (decimal 10) to register  X
.....
```

Storage

Features of programmable storage as per ISA:

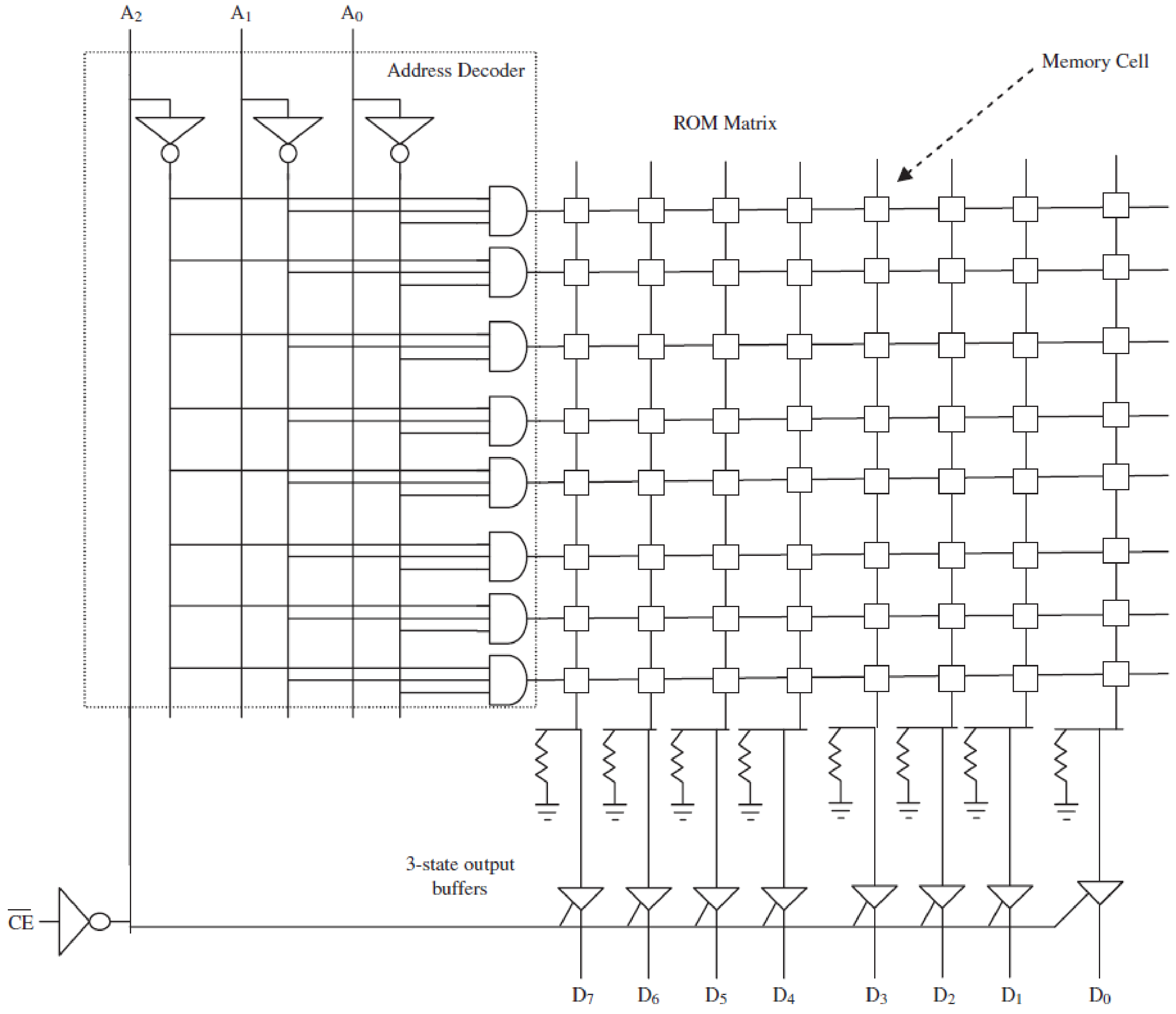
1. *The organization of memory used to store operands:*
    - Memory – array of storage that stores data
    - Indices – memory addresses
    - Actual range of addresses available to processor – address space
    - Characteristics:
        - Linear
        - Segmented
    - ISA defines where data is stored and how it is stored in memory.
        - Eg. Big-endian or little endian

2. Register Set

- A register is simply fast programmable memory normally used to store operands that are immediately or frequently used.
- A processor’s set of registers is commonly referred to as the ***register set*** or the ***register file.***
- *Depends on processor – register size will vary*

3. How Registers are used

- ISA defines which register can be used for what transactions.
  - Eg.: Special purpose, floating point, general purpose


A2
A1
A0
Address Decoder
Memory Cell
ROM Matrix
3-state output buffers
CE
D7
D6
D5
D4
D3
D2
D1
D0

Addressing Modes

- Addressing modes define how the processor can access operand storage.
- Memory addressing modes define usage of memory and registers.
- Two common types of addressing mode models:
  - **Load-Store Architectures**: only allows operations to process data in registers, not anywhere else in memory.
  - Eg.: PowerPC: load and store instructions: register plus displacement

  - **Register-Memory Architectures:** allows operations to be processed both within register and other types of memory.
  - Eg.: Intel’s i960 Jx processor: based upon the register-memory model (absolute, register-indirect)

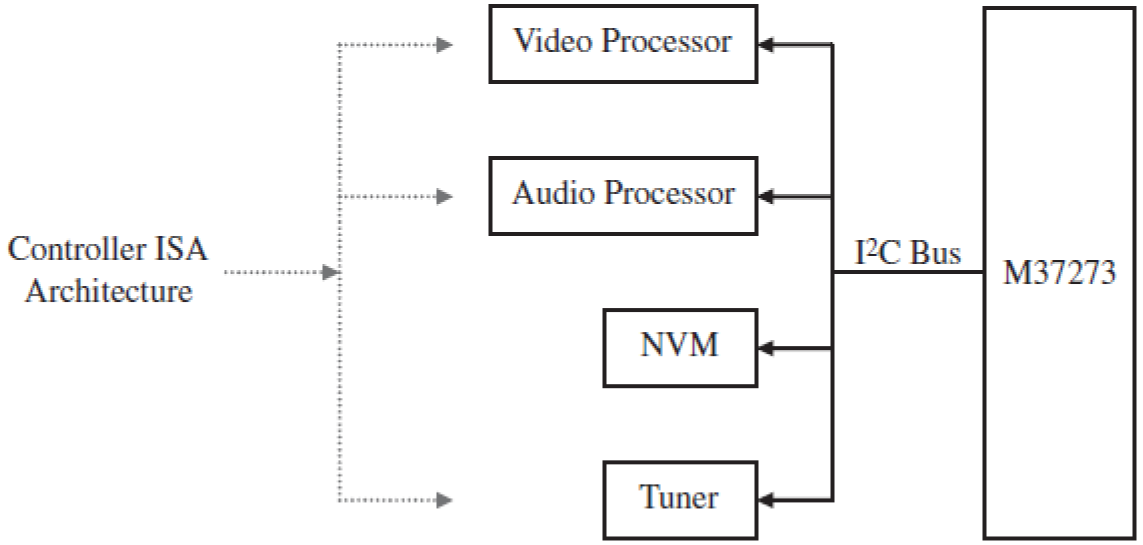
Interrupts and Exception Handling

- Interrupts: mechanisms that stop the standard flow of the program.
- Execute another set of code in response to some event, such as problems with the hardware, resets, and so forth

Application-Specific ISA Models

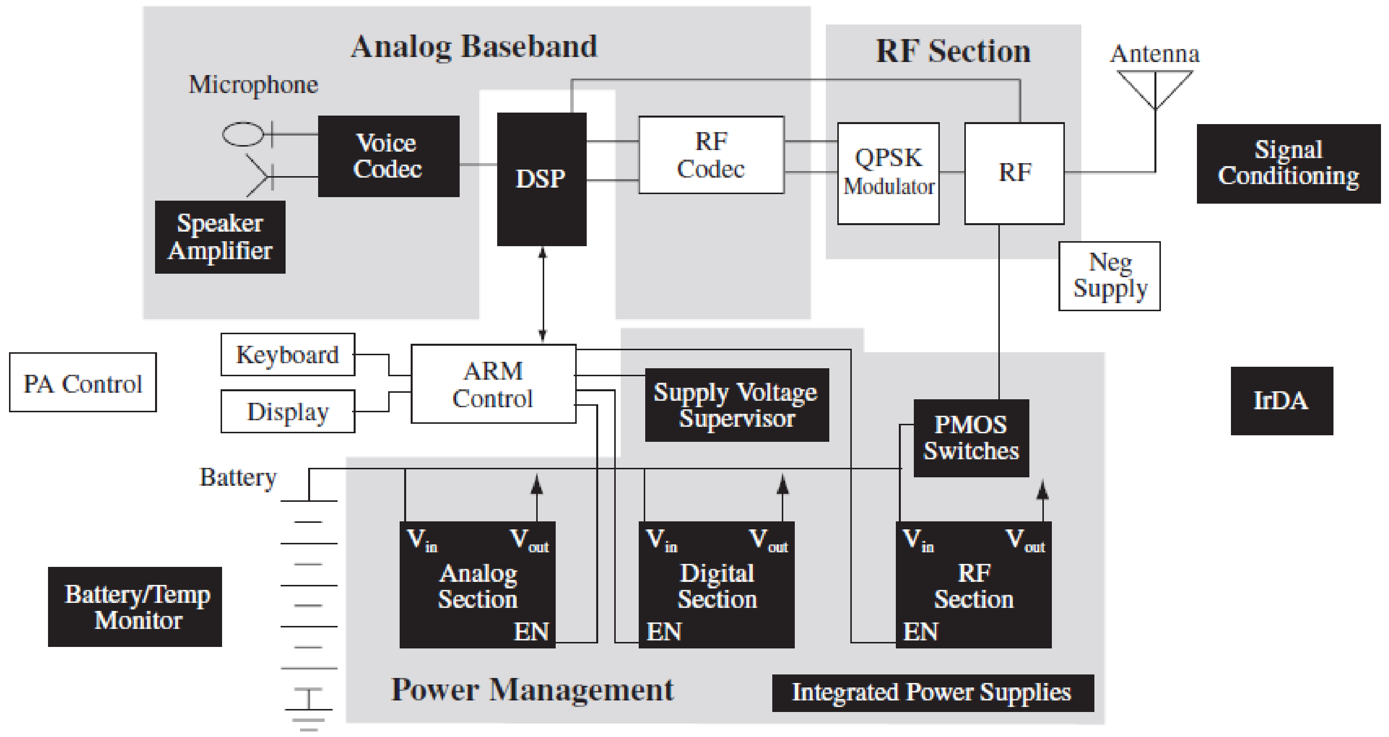
Controller Model

- Implemented in processors that are not required to perform complex data manipulation,
- Example-video and audio processors (used as slave processors on a TV board)



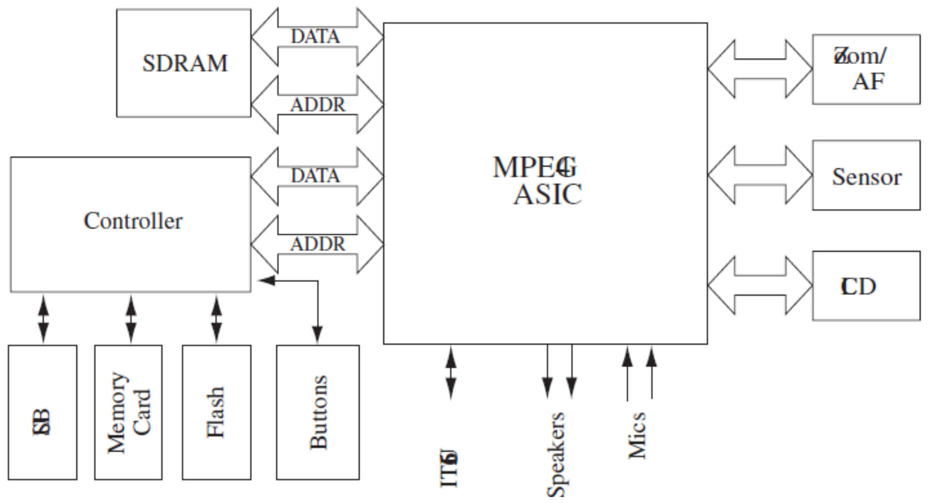
Datapath Model

- implemented in processors whose purpose is to *repeatedly perform fixed computations* on different sets of data.
- Eg. DSP processors



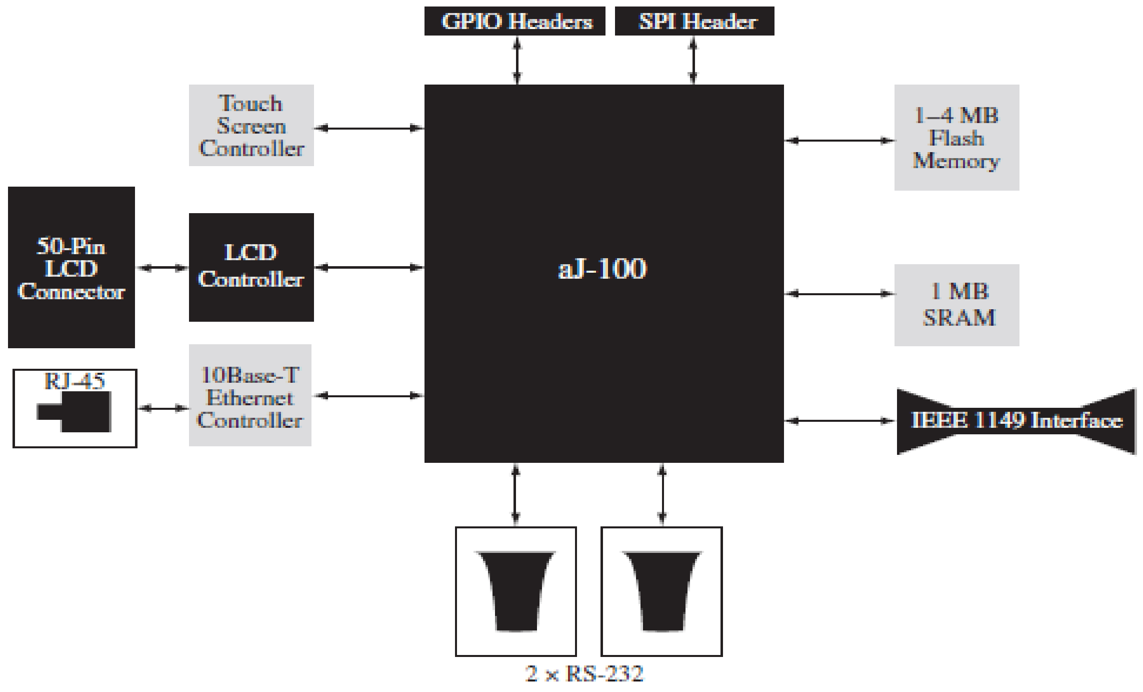
Finite State Machine with Datapath (FSDM) Model

- Combination of the Datapath ISA and the Controller ISA
- Used for processors that are not required to perform complex data manipulation and must repeatedly perform fixed computations on different sets of data.
- Eg. - application-specific integrated circuits (ASICs), programmable logic devices (PLDs), and field-programmable gate-arrays (FPGAs).



Java Virtual Machine (JVM) Model

- The JVM ISA is based upon one of the Java Virtual Machine standards (*Sun Microsystem's Java Language)*.
- Real-world JVMs can be implemented in an embedded system via hardware, such as in aJile's aj-80 and aj-100 processors,



General Purpose ISA Models

Complex Instruction Set Computing (CISC) Model

- Defines complex operations made up of several instructions.
- CISCs typically have multiple-cycle operations.
- Examples of architectures that implement a CISC ISA –
  - Intel's x86 and Motorola/Freescale's 68000 families of processors.

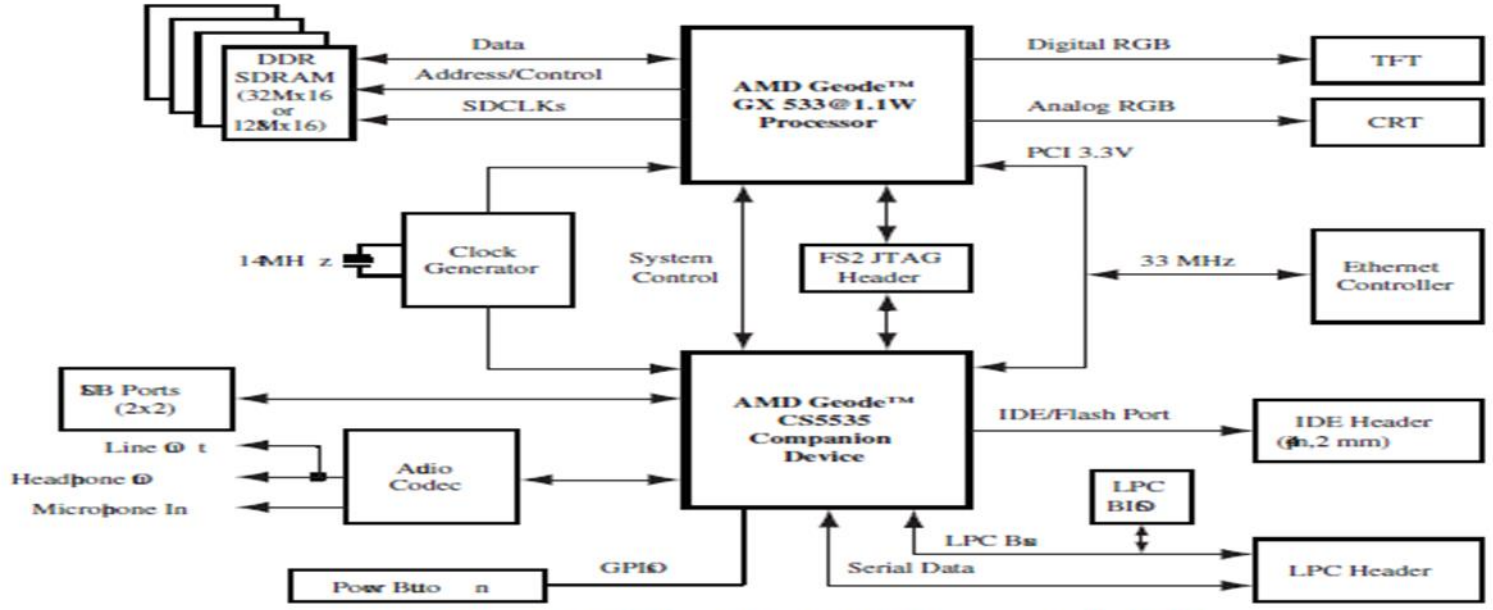


Figure 4-11: CISC ISA implementation example [4-10]

Reduced Instruction Set Model (RISC) Model

- Architecture with simpler and/or fewer operations made up of fewer instructions.
- architecture that has a reduced number of cycles per available operation.
- Many RISC processors have only one-cycle operations,
- Eg.:- ARM, PowerPC, SPARC, and MIPS

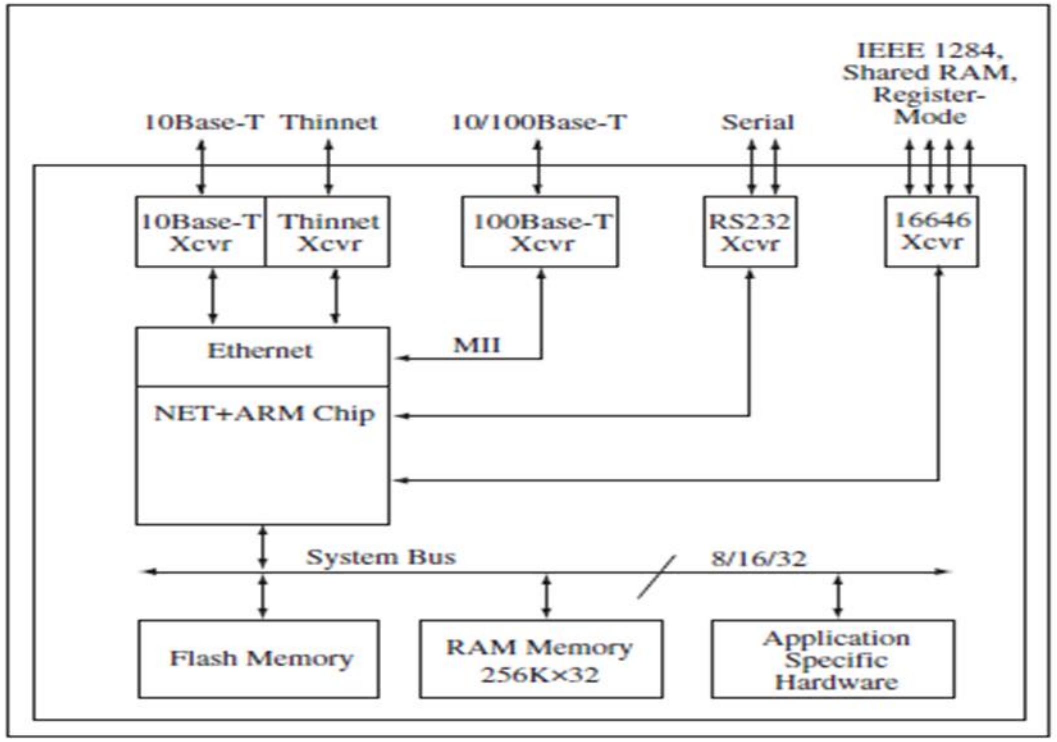


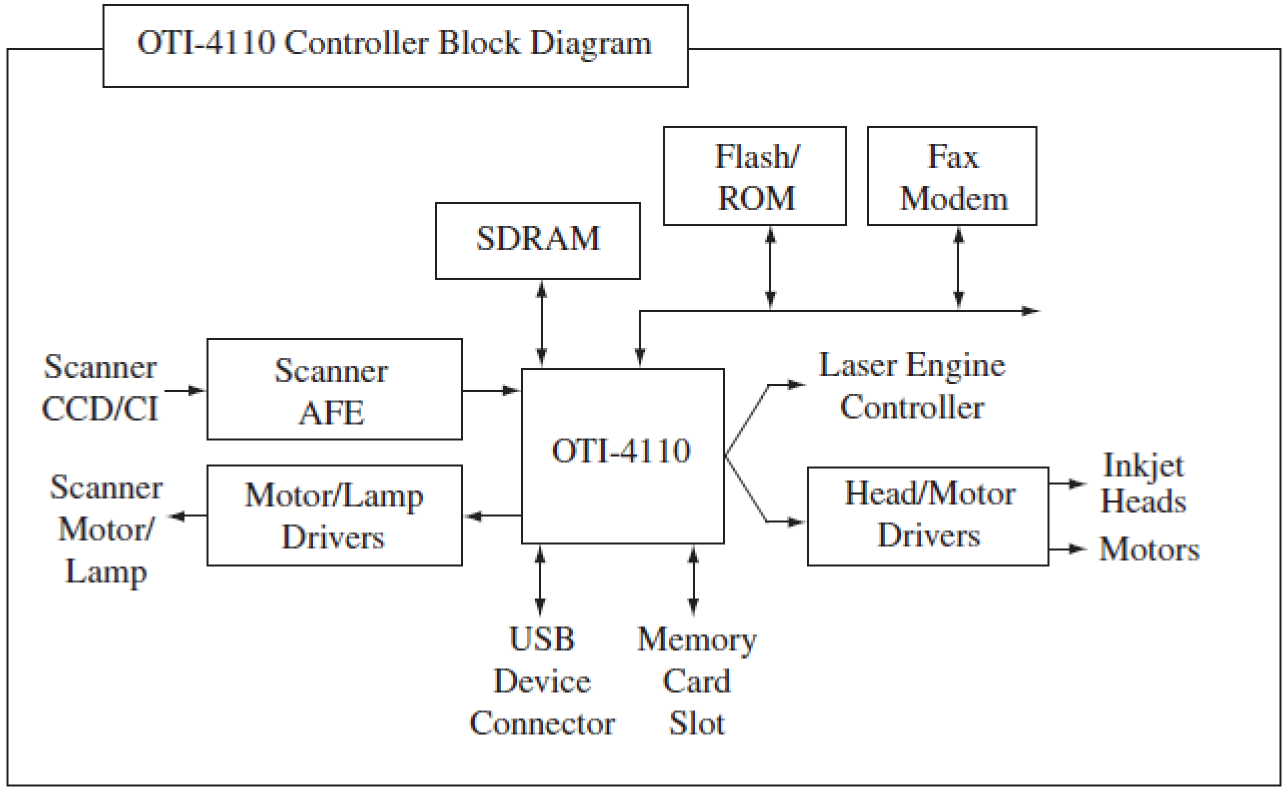
Figure 4-12: RISC ISA implementation example [4-11]

Instruction-Level Parallelism ISA Models

- These ISA architectures **execute multiple instructions in parallel**, as the name implies.
- Considered as higher evolutions of the RISC ISA.
- Example-
    - Single Instruction Multiple Data (SIMD) Model,
    - Superscalar Machine Model,
    - Very Long Instruction Word Computing (VLIW) Model

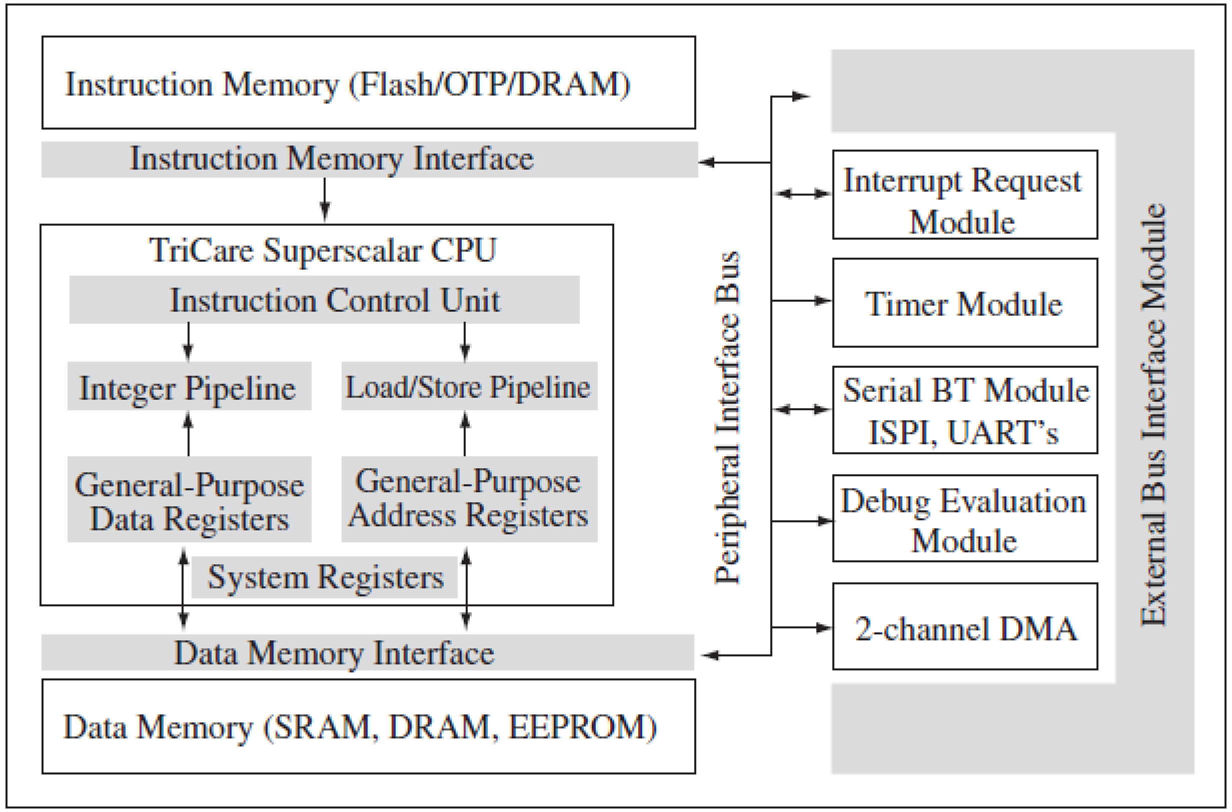
Single Instruction Multiple Data (SIMD) Model

- Designed to process an instruction simultaneously on multiple data components that require action to be performed on them.



Superscalar Machine Model

Able to process multiple instructions simultaneously within one clock cycle through the implementation of multiple functional components within the processor


Instruction Memory (Flash/OTP/DRAM)
Instruction Memory Interface
TriCare Superscalar CPU
Instruction Control Unit
Integer Pipeline
Load/Store Pipeline
General-Purpose Data Registers
General-Purpose Address Registers
System Registers
Data Memory Interface
Data Memory (SRAM, DRAM, EEPROM)
Peripheral Interface Bus
Interrupt Request Module
Timer Module
Serial BT Module ISPI, UART's
Debug Evaluation Module
2-channel DMA
External Bus Interface Module

Very Long Instruction Word Computing (VLIW) Model

- The VLIW ISA defines an architecture in which a very long instruction word is made up of multiple operations.
- These operations are then broken down and processed in parallel by multiple execution units within the processor.

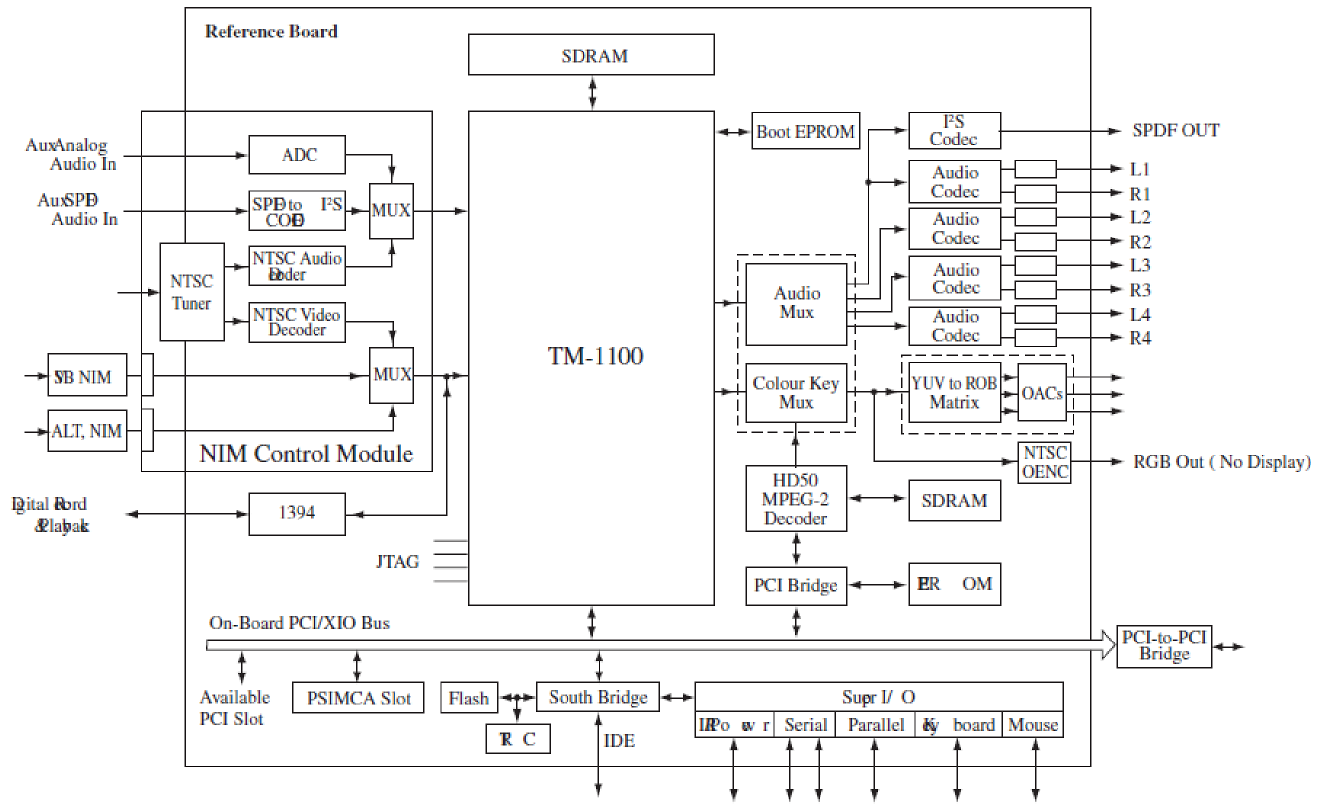


Figure 4-15: VLIW ISA implementation example—(VLIW) Trimedia-based DTV board [4-14]

Internal Processor Design

- The ISA defines ***what*** a processor can do, and it is the processor's internal interconnected hardware components that physically implement the ISA's features.
- Components of ISA <=> Components of embedded board
    - CPU
    - Memory
    - I/O components
    - Buses

Von-Neumann vs Harvard Architectures

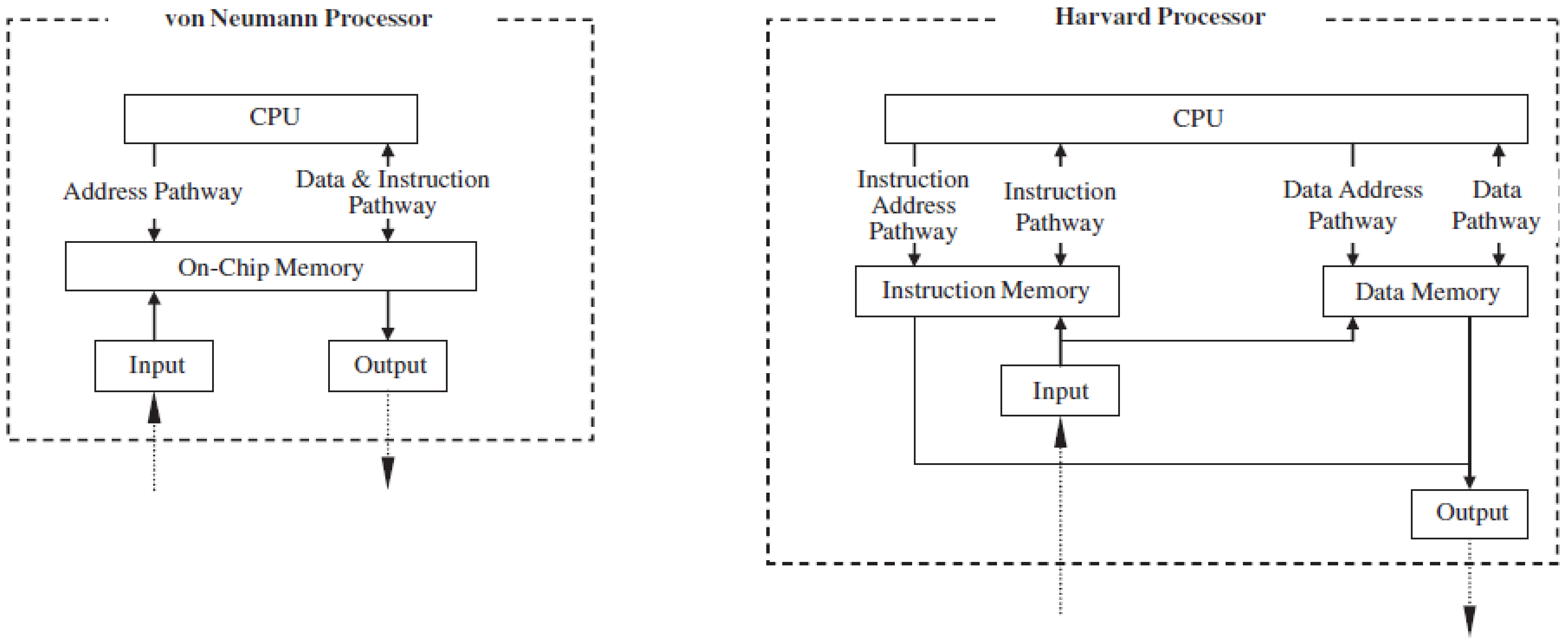
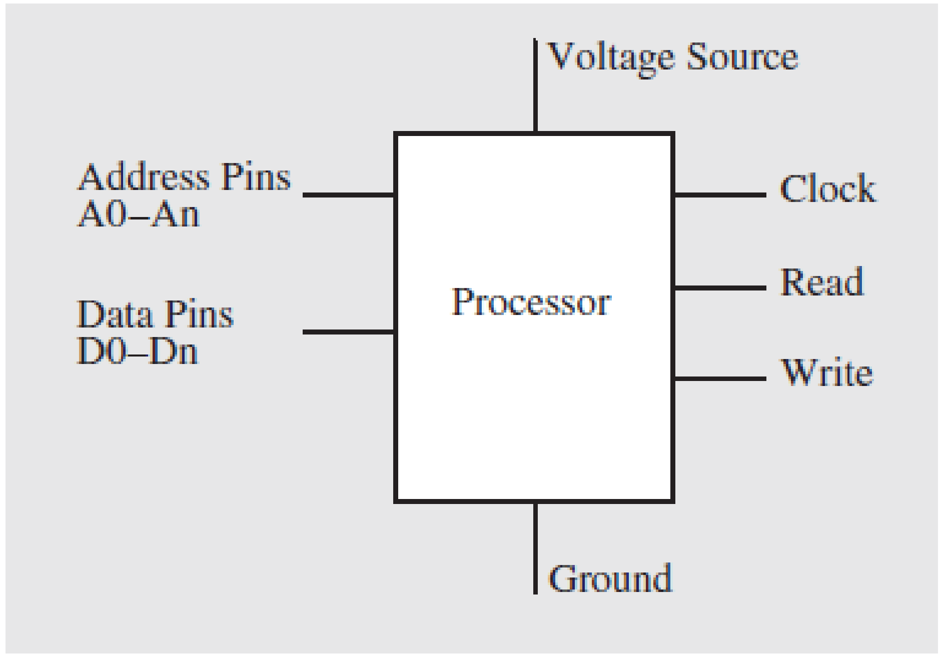


Figure 4-17: Von Neumann vs. Harvard architectures

Von-Neuman & Processor Pins

- Processors typically have address and data signals to read and write data to and from memory.
- In order to communicate to memory or I/O, a processor usually has some type of READ and WRITE pins to indicate it wants to retrieve or transmit data.



Central Processing Unit (CPU)

- the *processing unit* within a processor.
- The CPU is responsible for executing the cycle of fetchin decoding, and executing instructions.

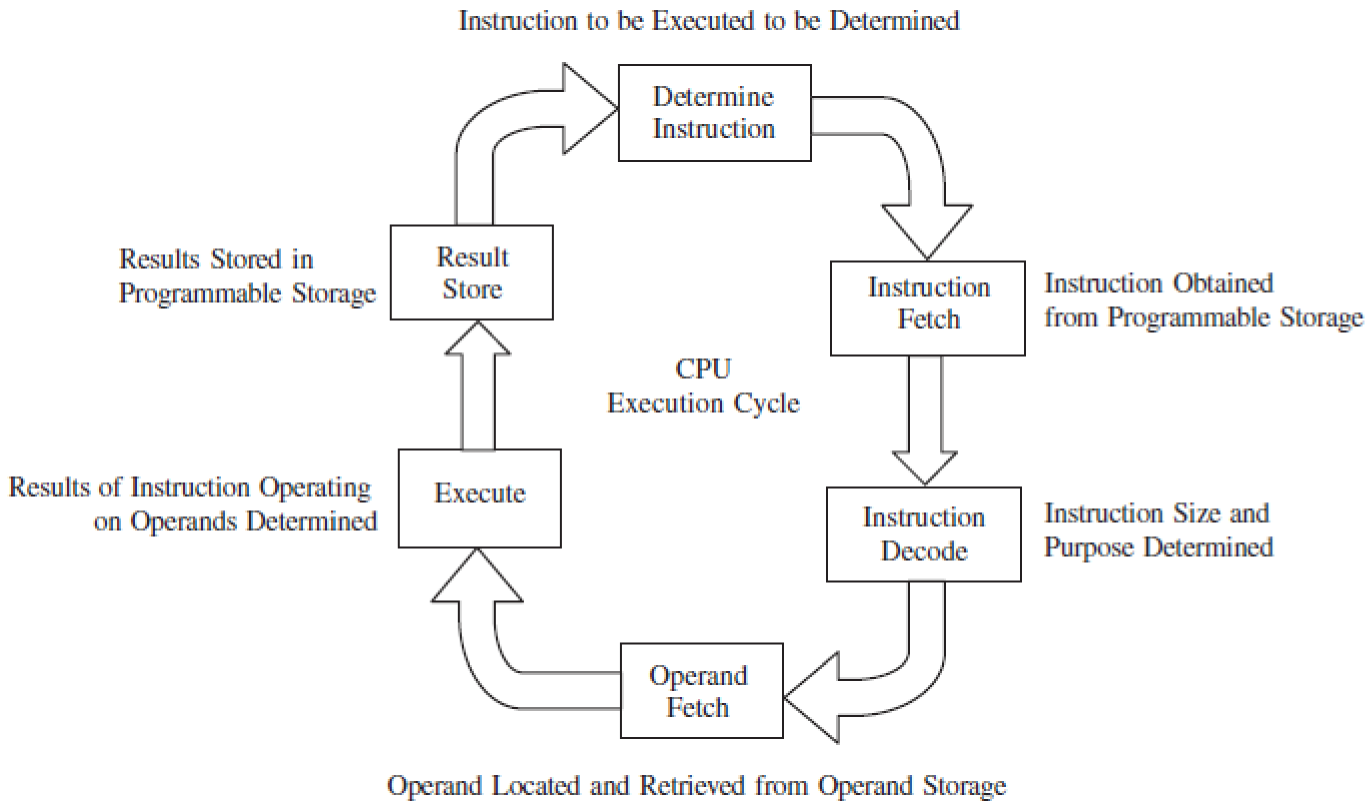

Instruction to be Executed to be Determined
Determine Instruction
Instruction Fetch
Instruction Obtained from Programmable Storage
Instruction Decode
Instruction Size and Purpose Determined
Operand Fetch
Operand Located and Retrieved from Operand Storage
Execute
Results of Instruction Operating on Operands Determined
Result Store
Results Stored in Programmable Storage
CPU Execution Cycle

Figure 4-20: Fetch, decode and execution cycle of CPU

CPU- von-Neumann

CPU components

- the arithmetic logic unit (ALU) – *implements the ISA's operations*
- registers – *a type of fast memory*
- the control unit (CU) – *manages the entire fetching and execution cycle*
- the internal CPU buses – *interconnect the ALU, registers, and the CU*

Internal CPU buses

- mechanisms that interconnect the CPU’s other components: the ALU, the CU, and registers.
- Bus types:
  - *Data bus* – carries data, bi-directionally, between registers and the ALU
  - *Address bus* - which carries the locations of the registers that contain the data to be transferred
  - *Control bus* - which carries control signal information, such as timing and control signals

ALU

- The arithmetic logic unit (ALU) implements the comparison, mathematical and logical operations defined by the ISA.
- ALU is responsible for accepting **multiple *n*-bit binary operands** and **performing** any logical (AND, OR, NOT, etc.), mathematical (+, –, *, etc.), and comparison (=, <, >, etc.) **operations on these operands.**
- Basic building block - Full-adders

Registers

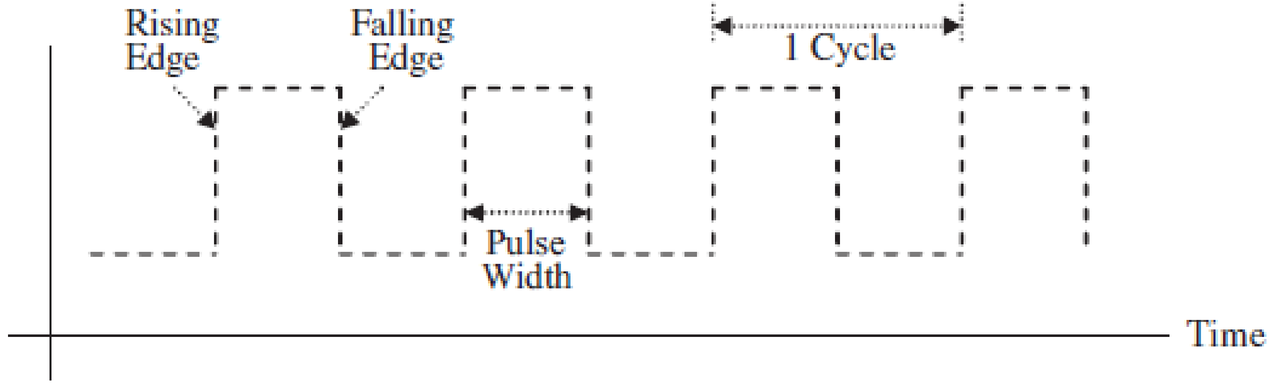
- Registers are simply a combination of various flip-flops that can be used to temporarily store data or to delay signals.
  - **Storage register -** fast programmable internal processor memory usually used to temporarily store, copy, and modify operands
  - **Shift register -** delay signals by passing the signals between the various internal flip-flops with every clock pulse
- *the number of flip-flops in each register* that is actually used to describe a processor → Eg.: 32 bit register size = 32 flipflop = 32 bit processor
- General purpose & special purpose – using flags

Control Unit

- The control unit (CU) is primarily responsible for generating timing signals.
- controlling and coordinating the fetching, decoding, and execution of instructions in the CPU.

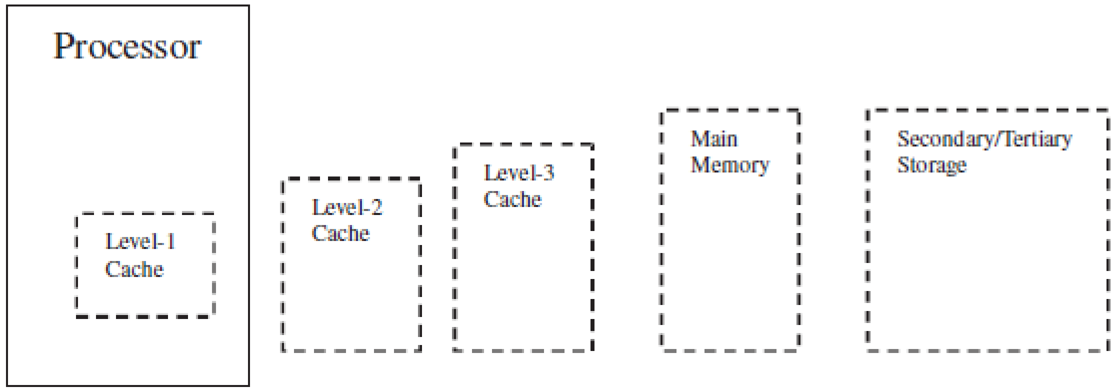
CPU & Master (System) clock

- A processor's execution is ultimately synchronized by an external *system* or *master clock,* located on the board.
- Fixed frequency sequence of regular on/off pulse signals (square waves).



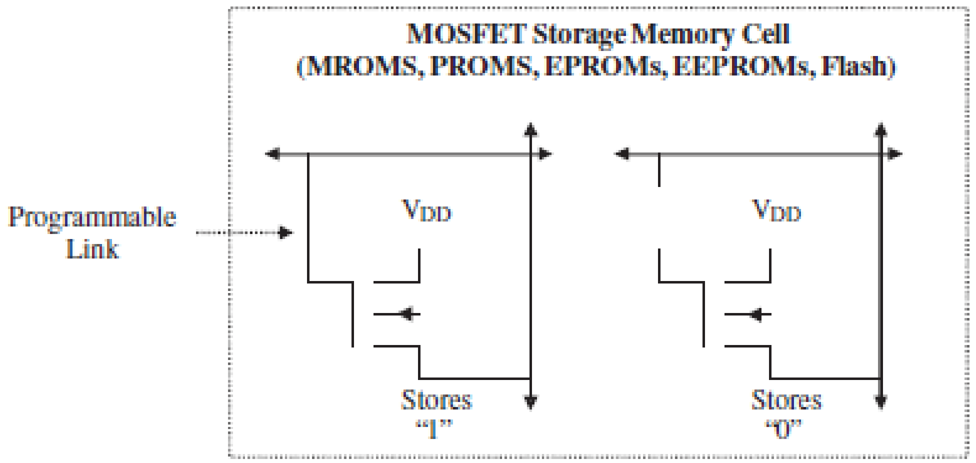
On-Chip Memory

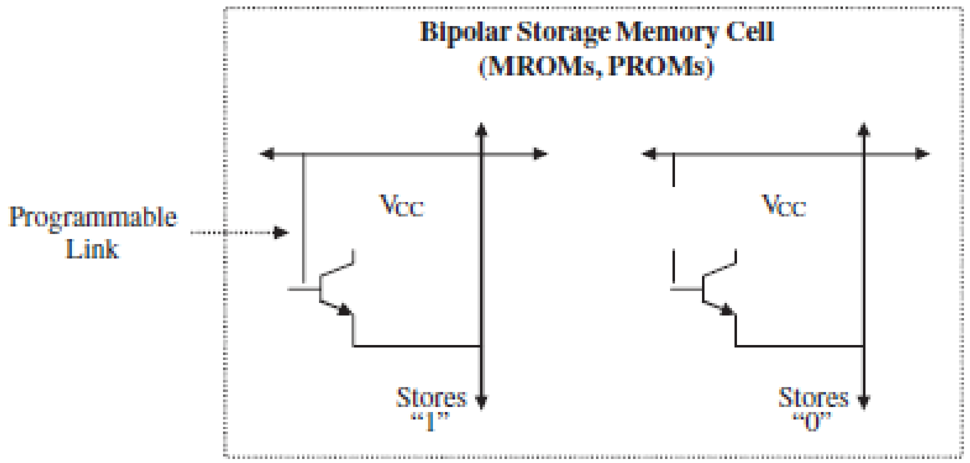
- Embedded platforms have a *memory hierarchy*, a collection of different types of memory, each with unique speeds, sizes, and usages.



ROM

- On-chip ROM is memory integrated into a processor that contains data that remain even when there is no power in the system.
- Have small, long-life battery
- Non-volatile memory.
- Stores data in memory cells


MOSFET Storage Memory Cell
(MROMS, PROMS, EPROMs, EEPROMs, Flash)
Programmable Link
VDD
VDD
Stores "1"
Stores "0"



Processor Performance

- Amount of work CPU completes in given time – **throughput**
- Processor's execution – synchronized with master clock
- Clock cycles/clock rate - > to control and coordinate with fetching, decoding and execution of instructions.
- CPU's clock rate is expressed in MHz.
- CPU's execution time → Total time the processor takes to process some program in seconds.
- length of time a CPU takes to complete a clock cycle = inverse of the clock rate (1/clock rate) → clock period or cycle time
- Unit of CPC – seconds per cycle

CPI (Clock cycles per Instruction)

CPI = Σ(CPI per instruction * instruction frequency)

- ***CPU execution time in seconds per program***

  *=(total number of instructions per program or instruction count) * (CPI in number of cycle cycles/instruction) * (clock period in seconds per cycle)*

  *= ((instruction count) * (CPI in number of cycle cycles/instruction)) / (clock rate in MHz)*

- **CPU thorughput** *(in bytes/sec or MB/sec)*

  = 1 / CPU execution time = CPU performance

Other measures

- ***Latency:*** length of elapsed time a processor takes to respond to some event.
- ***Availability:*** the amount of time the processor normally runs without failure
- ***Reliability:*** the average time between failures or MTBF (mean time between failures)
- ***Recoverability:*** the average time the CPU takes to recover from failure or MTTR (mean time to recover).

MIPS and Myths!

- MIPS = Instruction Count / (CPU execution time * 106)

  = Clock Rate / (CPI * 106)
- Faster processor – higher MIPS?
- Instruction complexity and functionality aren't taken in the MIPS formula, So, MIPS cannot compare the capabilities of processors with different ISAs.
- MIPS can vary on the same processor when running different programs (with varying instruction counts and different types of instructions).
- Software programs called ***benchmarks*** can be run on a processor to measure its performance

Summary

- Embedded processor – what is made of
- Concepts of ISA
- Processor design – ISA
- Von Neumann model – components of embedded board
- Processor performance measures