R·I·T

NIOS II Processor

Reminders

- No quiz this weekend
- Assembly demo due Friday
- Lab 1 due next week
  - **Get a head start**
  - **Next lab will be challenging**

Last Weeks Questions

- What is a caller saved register?
  - **Volatile registers used to hold temporary values that need do not need to be preserved between calls**
  - **Call refers to a procedural access**
  - **Callee saved registers are non-volatile and must be saved if a function needs to overwrite the value**

- Can a stw offset be a register? No

stw / stwio

Instruction	store word to memory or I/O peripheral
Operation	Mem32[rA + σ(IMM16)] ← rB
Assembler Syntax	`stw rB, byte_offset(rA)` `stwio rB, byte_offset(rA)`
Example	`stw r6, 100(r5)`

Processor Types - CISC

- CISC Processor Architecture
  - **CISC ➔ Complex Instruction Set Computer**
    - Processors that are easy to program and make efficient use of memory.
    - Earliest machines were programmed in assembly language and memory was slow and expensive
      - ***Big programs require a lot of memory $$$$***
  - **CISC developed to make compiler development simpler**
    - Shifts burden of generating machine instructions to the processor
    - Ex: Instead of compiler writing long machine instructions to calculate a square-root →ability was built-in CISC processor
    - Instructions require multiple clock cycles to execute
  - **Examples:**
    - Intel 80x86
    - Motorola 68K series

CISC Processors

- CISC Instruction Sets
  - **More complex instructions**
    - Instruction fit into multiple words
    - More complex hardware
    - Shorter programs
  - **Complex address modes supported**
  - **Memory-to-memory transfer supported**

Processor Types - RISC

- RISC Processor Architecture
  - **RISC➔ Reduced Instruction Set Computer**
    - Processor architecture that uses small, highly-optimized set of instructions
  - **RISC Characteristics**
    - *one cycle execution time*
      - ***RISC processors have a CPI (clock per instruction executed)***
    - Pipelining
      - ***Technique for simultaneous execution of instructions***
    - Large number of registers
      - ***Prevent large interactions with memory***
  - **Examples:**
    - ARM
    - Nios II
    - PowerPC

RISC Processors

- RISC Instruction Sets
  - **Simpler instructions**
    - Instruction fits into single word
    - Simpler hardware
    - Faster instruction execution
    - Longer programs
  - **Simpler address modes**
  - **Load/Store architecture used**
    - Memory only access via load/store instructions
    - Arithmetic/logic operations work on register

Processor Types

R·I·T

CISC	RISC
Emphasis on hardware • Software simple	Emphasis on software • Hardware simple
Includes multi-clock, complex instructions	Single-clock, reduced instruction only
Memory-to-memory: "LOAD" and "STORE" incorporated in instructions	Register to register: "LOAD" and "STORE" are independent instructions
Small code sizes, high cycles per second	Low cycles per second, large code sizes
Transistors used for storing complex instructions	Spends more transistors on memory registers

RISC systems shorten execution time by reducing the *clock cycles per instruction (i.e. simple instructions take less time* to interpret)

CISC systems shorten execution time by reducing the *number of instructions per program*

Processor Types

- CISC and RISC Processor Architecture
  - **Line between RISC and CISC blurring**
    - CISC processor borrowing goodness from RISC
      - ***Ex: Pipeline introduced into CISC***
    - RISC processor borrowing goodness from CISC
      - ***Ex: More complex addressing added into RISC***

Nios II Processor Overview

- What is the Nios II Processor?
  - **Soft-core processor developed by Altera (now Intel)**
    - Soft core design ➔ not fixed in silicon
    - Soft core allows customizing of processor and peripherals
      - ***Add processor features as required***
      - ***Remove unused processor features***
      - ***Create custom instructions***
  - **Nios II is second generation of software processor**
  - **32-bit RISC Processor**
    - RISC ➔Simple instructions provide higher performance
      - ***If instructions are simple enough to execute very quickly***

NIOS II Basics

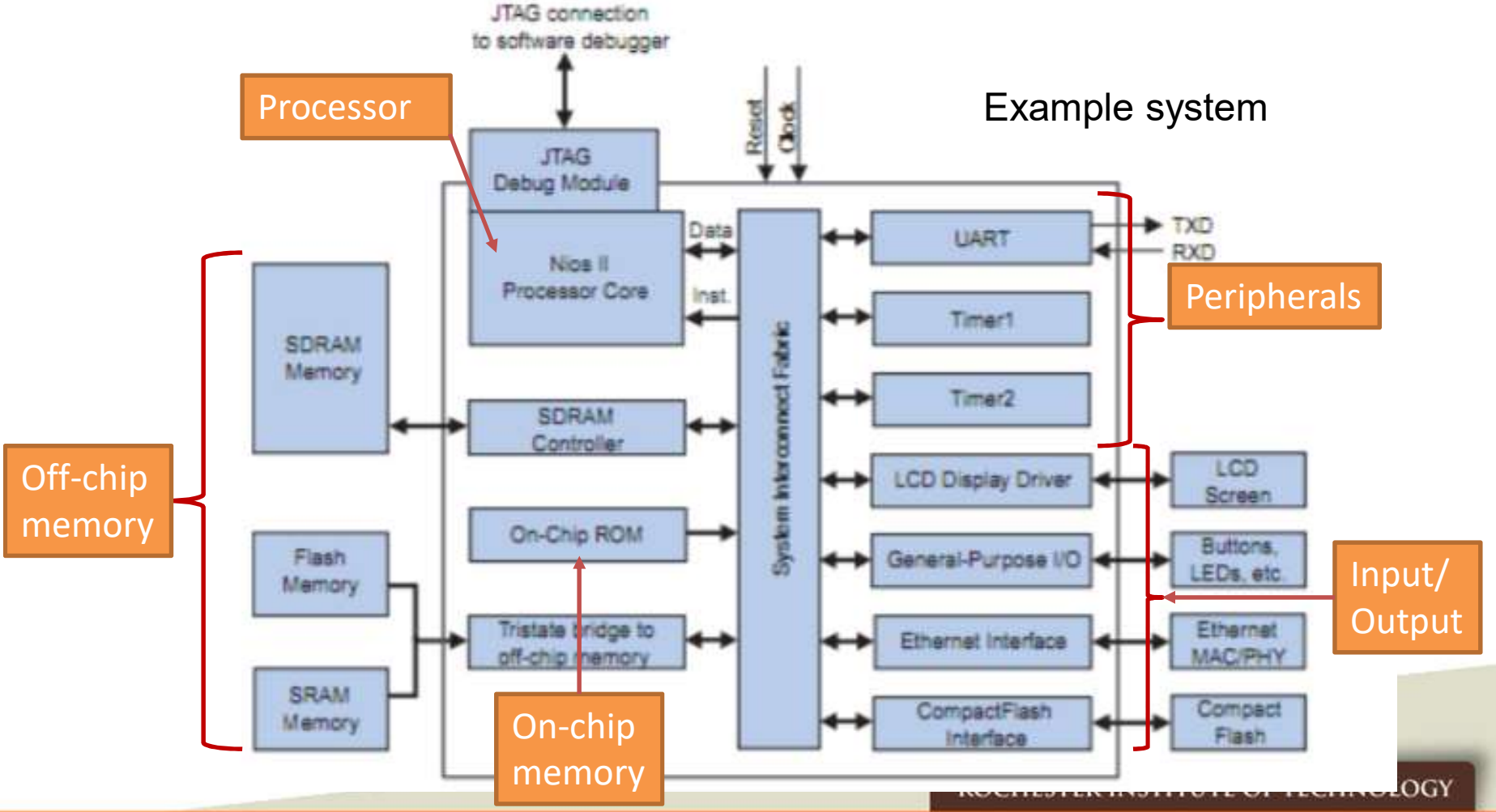
- General-purpose RISC processor core
  - **32-bit instruction set, data path and address space**
  - **32 general-purpose registers**
  - **Optional shadow register sets – useful for context switching and multitasking systems (RTOS)**
  - **32 interrupt sources**
  - **External interrupt controller interface for more interrupt sources**
  - **Single instruction 32 x 32 multiply and divide producing a 32 bit result**
  - **Dedicated instructions for computing 64-bit and 128-bit products of multiplication**
  - **Floating-point instructions for single-precision floating-point operations**

NIOS II Basics (con't)

- **Single-instruction barrel shifter**
- **Access to a variety of on-chip peripherals, and interfaces to off-chip memories and peripherals**
- **Optional memory management unit (MMU) to support operating systems that require MMU**
- **Optional memory protection unit (MPU)**
- **Instruction set architecture (ISA) compatible across all NIOS II processor systems**

NIOS II Processor System

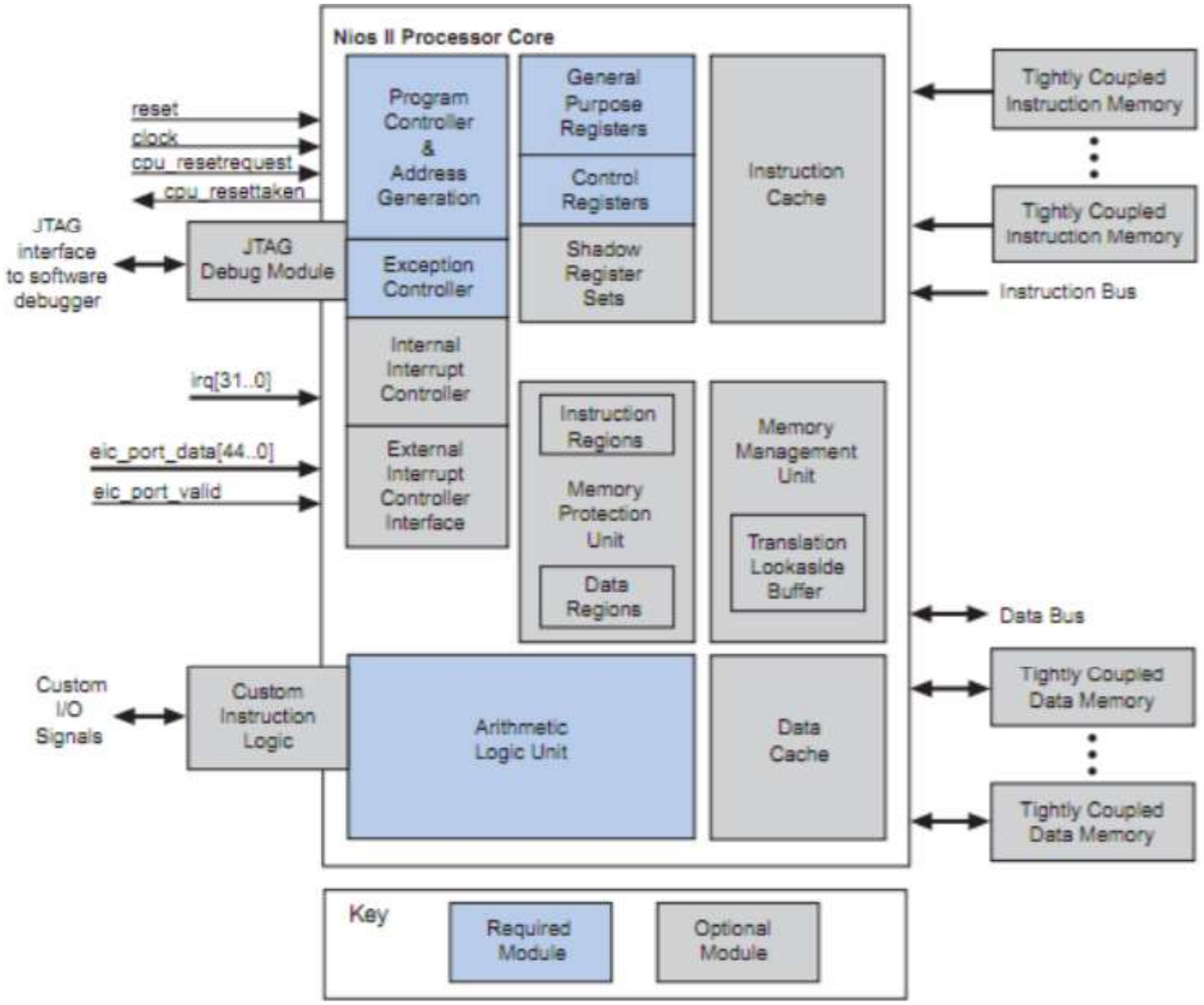
- A NIOS II processor system is equivalent to a microcontroller that includes a processor, memory and a combination of peripherals



NIOS II Architecture

- The NIOS II architecture defines the following functional units:
  - **Register file**
  - **Arithmetic logic unit (ALU)**
  - **Interface to custom instruction logic**
  - **Exception controller**
  - **Internal or external interrupt controller**
  - **Instruction bus**
  - **Data bus**
  - **Memory management unit (MMU)**
  - **Memory protection unit (MPU)**
  - **Instruction and data cache memories**
  - **Tightly-coupled memory interfaces for instruction and data**
  - **JTAG debug module**

NIOS II Architecture



NIOS II Architecture

- The functional units of the NIOS II architecture form the foundation for the NIOS II instruction set
  - **This does not indicate that any unit is implemented in hardware**
  - **The NIOS II architecture describes an instruction set, not a particular hardware implementation**
  - **A functional unit can be implemented in hardware, emulated in software or omitted entirely**
- A NIOS II implementation is a set of design choices embodied by a particular NIOS II processor core
  - **Each implementation achieves specific objectives such as smaller core size or higher performance**
  - **Allows the NIOS II architecture to adapt to the needs of different target applications.**

NIOS II Processor Tradeoffs

- Implementation variables generally fit one of three trade-off patterns:
  - **More or less features**
    - Example: To fine-tune performance, you can increase or decrease the amount of instruction cache memory. A larger cache increases execution speed of large programs, while a smaller cache conserves on-chip memory resources
  - **Inclusion or exclusion of a feature**
    - Example: To reduce cost, you can choose to omit the JTAG debug module. This decision conserves on-chip logic and memory resources, but it eliminates the ability to use a software debugger to debug applications.
  - **Hardware implementation or software emulation of a feature**
    - Example: For applications that rarely perform complex arithmetic, you can choose for the division instruction to be emulated in software. Removing the divide hardware conserves on-chip resources but increases the execution time of division operations

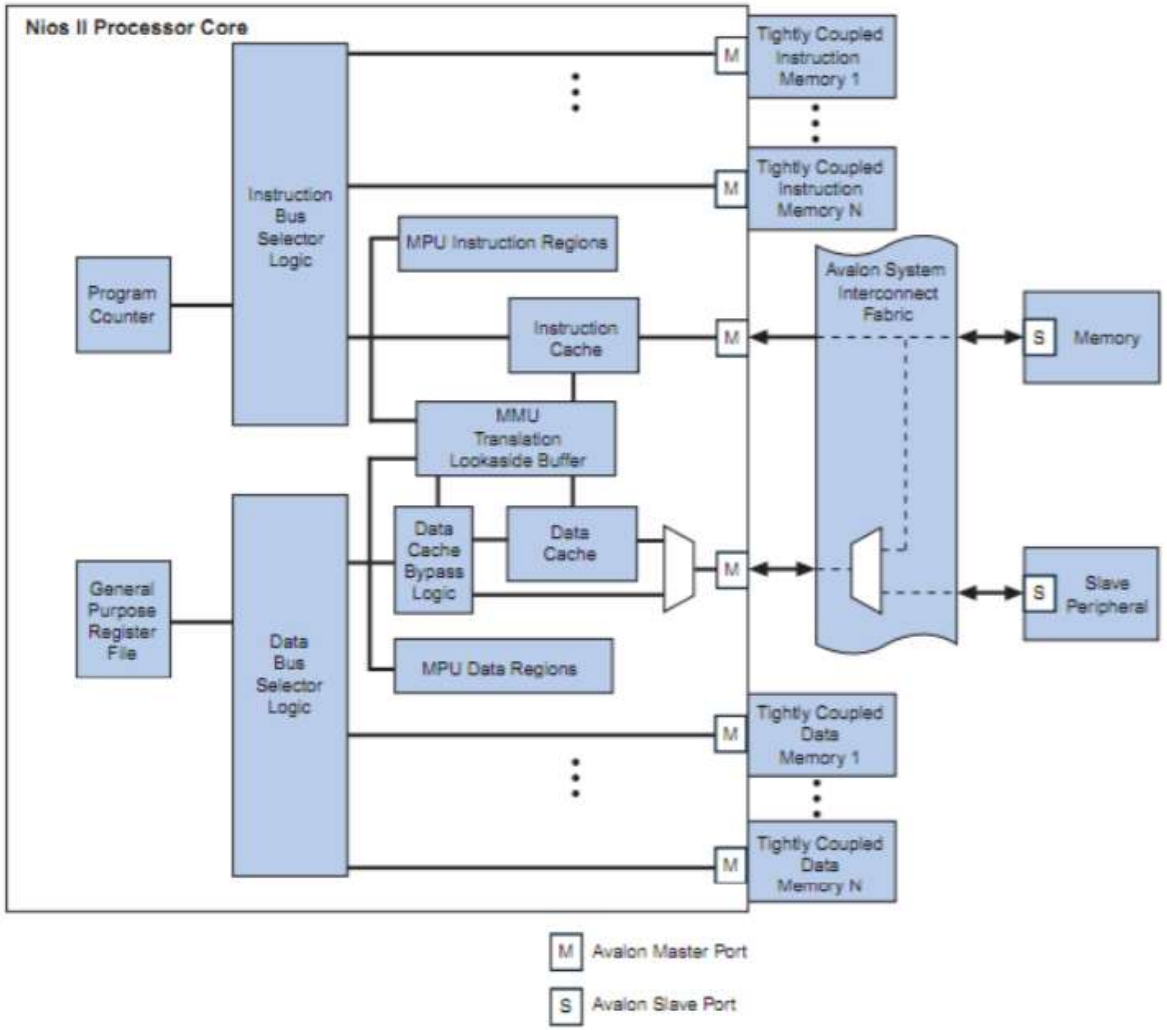
Memory and I/O Organization

- The flexible nature of the NIOS II memory and I/O organization are the most notable difference between NIOS II processor systems and traditional microcontrollers
- Because NIOS II processor systems are configurable, the memories and peripherals vary from system to system
  - **As a result, the memory and I/O organizations varies from system to system**

Memory and I/O Organization

- A NIOS II core uses one or more of the following to provide memory and I/O access:
  - **Instruction master port** – An Avalon Memory-Mapped (Avalon-MM) master port that connects to instruction memory via system interconnect fabric
  - **Instruction cache –** Fast cache memory internal to the NIOS II core
  - **Data master port** – An Avalon-MM master port that connects to data memory and peripherals via system interconnect fabric
  - **Data cache** – Fast cache memory internal to the NIOS II core
  - **Tightly coupled instruction or data memory port** – interface to fast on-chip memory outside the NIOS II core
- The NIOS II architecture hides the hardware details from the programmer so programmers can *usually* develop applications without specific knowledge of the hardware implementation

Memory and I/O Organization



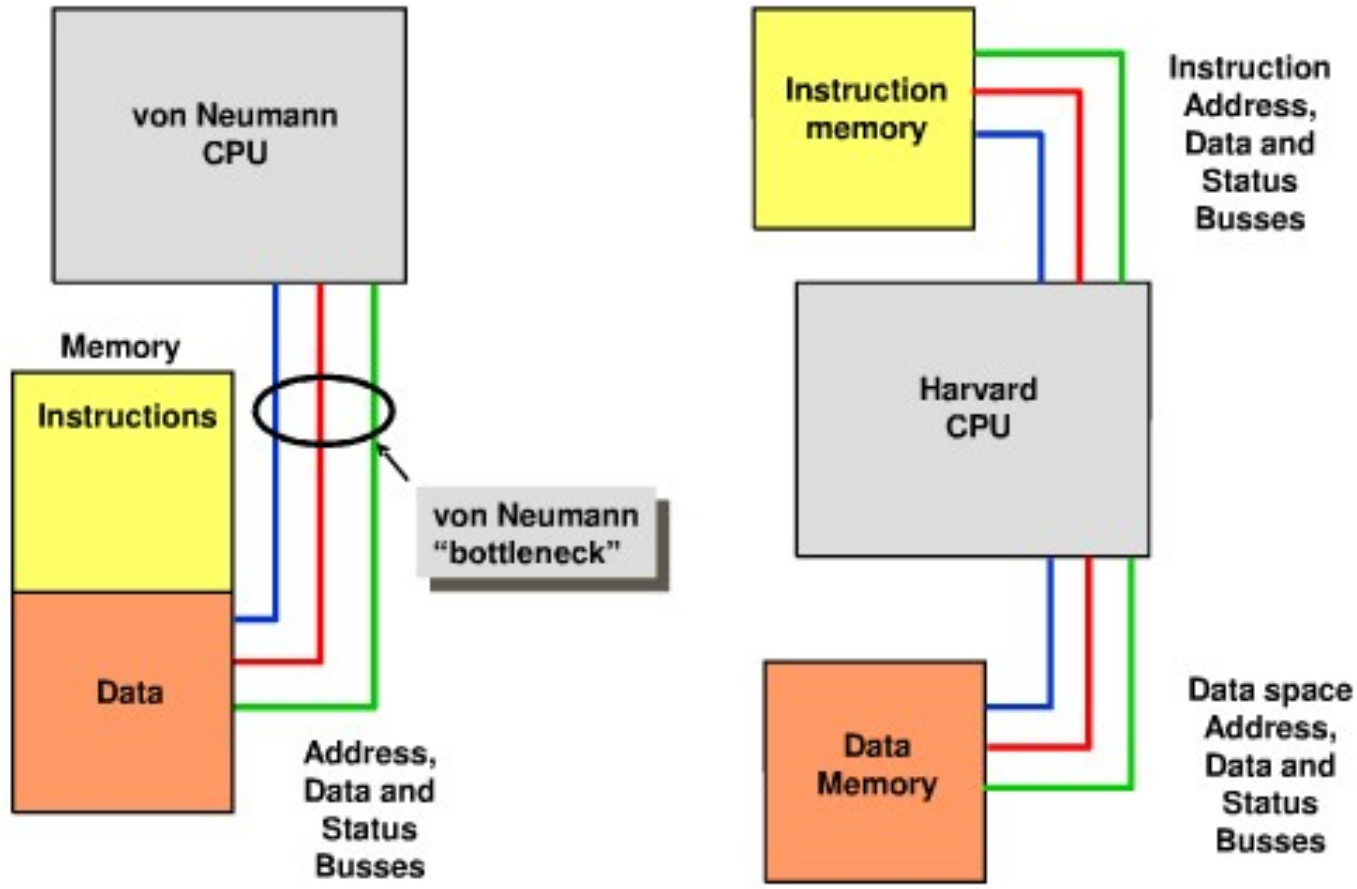
Memory and I/O Organization

- The NIOS II architecture does not specify anything about the existence of memory and peripherals; the quantity, type, and connection of memory and peripherals are system-dependent
  - **Typically, NIOS II processor systems contain a mix of fast on-chip memory and slower off-chip memory**
  - **Peripherals typically reside on-chip, although interfaces to off-chip peripherals also exist**

Two Types of Processor Architectures

- Harvard
  - **Has physically separate signals and storage for code and data memory**
  - **Two data buses and two address buses**
  - **Instructions can be fetched simultaneously with data being fetched**
- VonNeumann
  - **Shared signals and memory for code and data**
  - **One data bus and one address bus**
  - **Program can be modified by itself**

Two Types of Processor Architectures

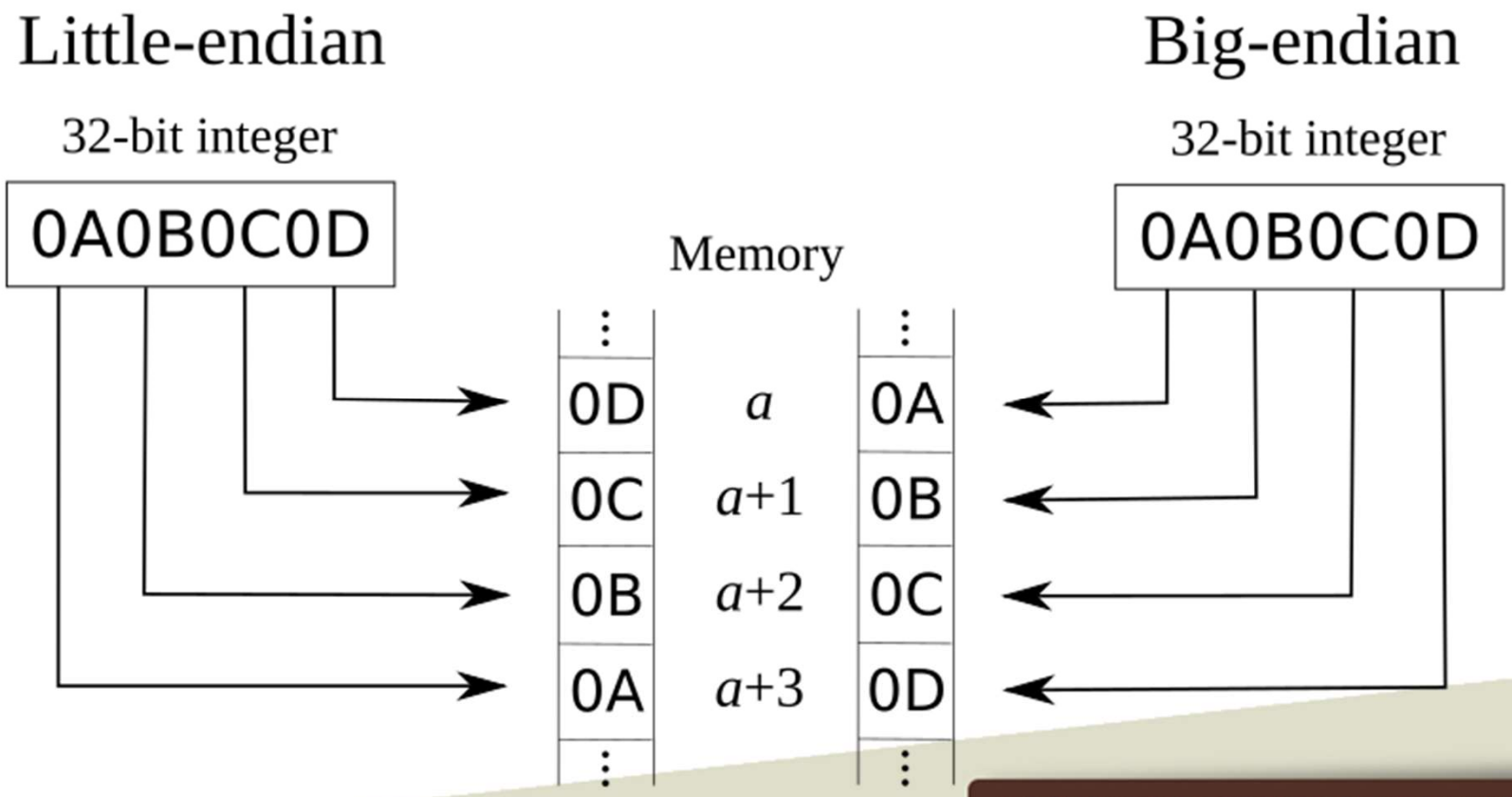


Instruction and Data Buses

- The NIOS II architecture supports separate instruction and data buses, classifying it as a Harvard Architecture
  - **Both the instruction and data buses are implemented as Avalon-MM master ports that adhere to the Avalon-MM interface specification**
  - **The data master port connects to both memory and peripheral components, while the instruction master port connects only to memory components**
    - Known as *memory-mapped I/O* – both the memory and peripherals are mapped into the address space of the data master port
    - Use pointers to addresses in programs to access both memory and I/O

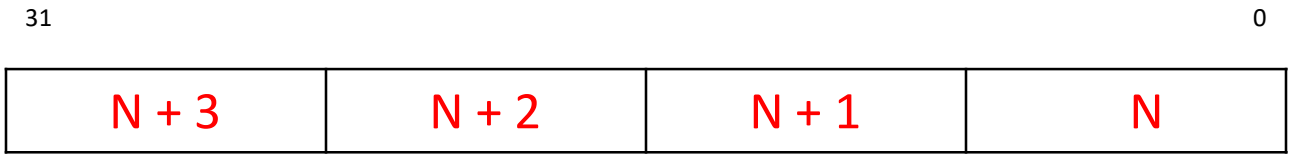
Two Types of Data Storage

- *Endian* describes the order in which bytes of data are stored in memory



Data Storage

- The NIOS II architecture is *little endian*
  - **Words and halfwords are stored in memory with the more-significant bytes at higher addresses**



- Words in NIOS II are 32 bits
  - **In ESD 1, when in doubt, the answer is 32!**

Nios II Processor Overview (Classic)

- Three levels of Performance
  - **Allows design scale as requirements change**

	Nios II /f Fast	Nios II /s Standard	Nios II /e Economy
Pipeline	6 Stage	5 Stage	None
H/W Multiplier & Barrel Shifter	1 Cycle	3 Cycle	Emulated In Software
Branch Prediction	Dynamic	Static	None
Instruction Cache	Configurable	Configurable	None
Data Cache	Configurable	None	None
Logic Requirements (Typical LEs)	1800 w/o MMU 3200 w/ MMU	1200	600
Custom Instructions	Up to 256		

Nios II Processor Overview (Classic)

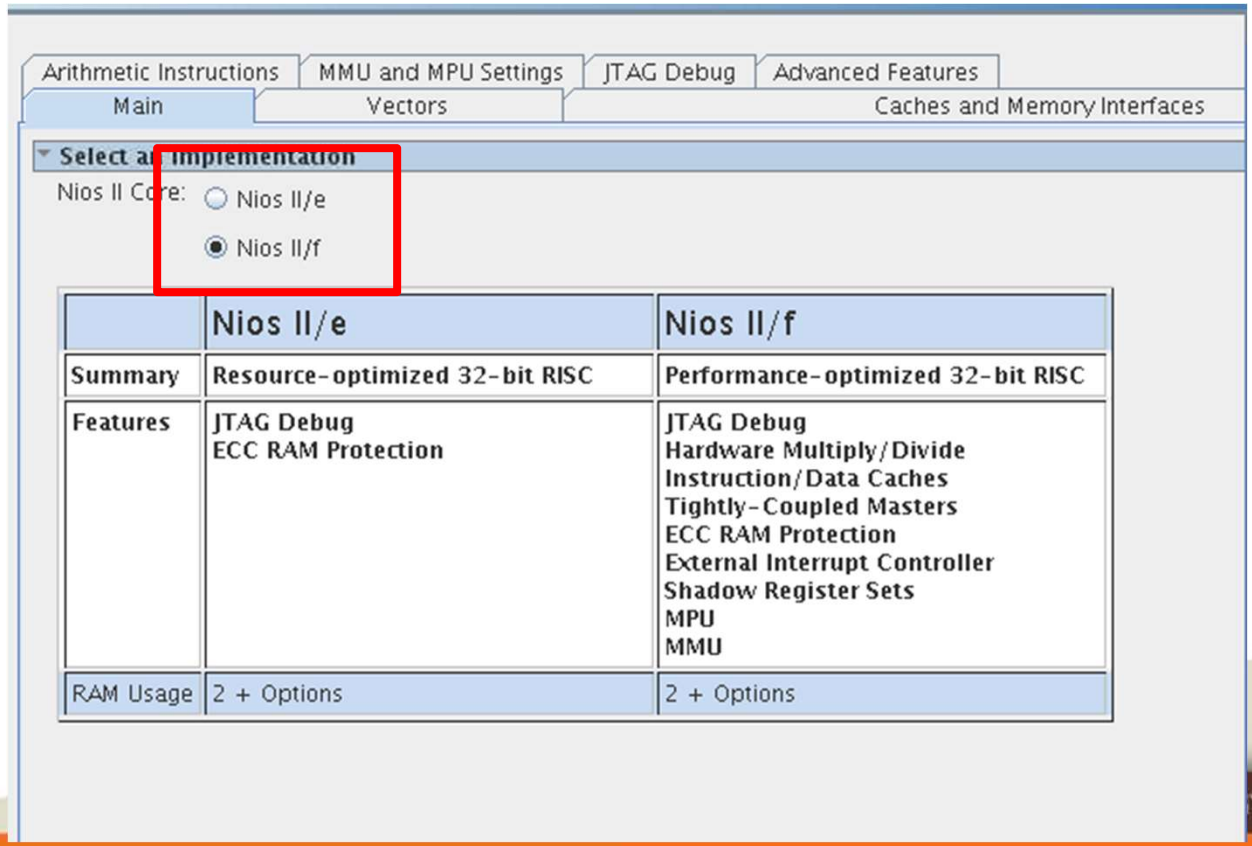
- Three level of Performance
  - **Performance & Size**
    - Performance varies by factor of 9
    - Size varies by factor of 3

Feature		Core		
		Nios II/e	Nios II/s	Nios II/f
Objective		Minimal core size	Small core size	Fast execution speed
Performance	DMIPS/MHz *(1)*	0.15	0.74	1.16
	Max. DMIPS *(2)*	31	127	218
	Max. f_{MAX} *(2)*	200 MHz	165 MHz	185 MHz
Area		< 700 LEs; < 350 ALMs	< 1400 LEs; < 700 ALMs	Without MMU or MPU: < 1800 LEs; < 900 ALMs With MMU: < 3000 LEs; < 1500 ALMs With MPU: < 2400 LEs; < 1200 ALMs
Pipeline		1 stage	5 stages	6 stages
External Address Space		2 GB	2 GB	2 GB without MMU 4 GB with MMU

NIOS II Processor Overview

- Our version of Platform Designer suggests not using classic
- We will choose between NIOS II/e and NIOS II/f

- **We will be using NIOSII/e this semester, but the default is NIOSII/f**
- Be sure to choose the correct processor!



	Nios II/e	Nios II/f
Summary	Resource-optimized 32-bit RISC	Performance-optimized 32-bit RISC
Features	JTAG Debug ECC RAM Protection	JTAG Debug Hardware Multiply/Divide Instruction/Data Caches Tightly-Coupled Masters ECC RAM Protection External Interrupt Controller Shadow Register Sets MPU MMU
RAM Usage	2 + Options	2 + Options