SEE3223 Microprocessors

1: Embedded Systems

Muhammad Mun’im Ahmad Zabidi (munim@utm.my)

BY NC SA

Microprocessor-Based Systems

- Aims
  - To review the main elements of a microprocessor system.
- Intended Learning Outcomes
  - At the end of this module, students should be able to:
    - Define and explain important terms associated with both hardware and software elements of a microprocessor system
    - Tell the difference between general purpose computing and embedded computing
    - List down the major components inside a computer & processor
    - Tell the difference between computer, processor, microprocessor and microcontroller
    - Explain instruction execution cycles of a generic microprocesso

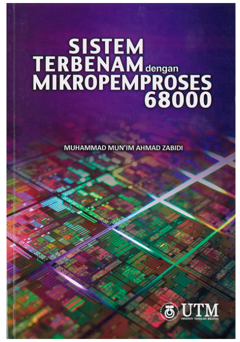
SEE3223 Microprocessor Systems

- What's in this course:
  - Assembly language programming
  - Microprocessor concepts
  - Hardware interfacing
- Pre-Requisites
  - Number representation, coding, registers, state machines
  - Realisation of simple logic circuits
  - Integrated circuit technologies
  - Designing with MSI components
  - Flip-Flops
  - Counters and sequential MSI components
  - Register transfer logic

Reading List

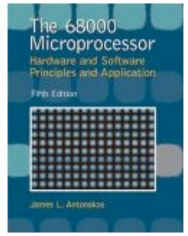
Required Text:

- Muhammad Mun'im Ahmad Zabidi (2011), *Sistem Terbenam dengan Mikropemproses 68000*, Penerbit UTM Press.

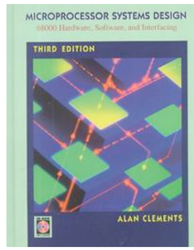


Recommended Readings:

- Antonakos, J.L. (2003), *The 68000 Microprocessor: Hardware and Software Principles and Applications, 5th Ed*., Prentice Hall.



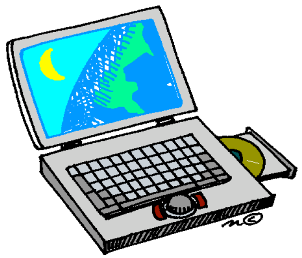
- Clements A.(1997), *Microprocessor Systems Design: 68000 Software, Hardware and Intefacing, 3rd Ed*., PWS Kent Publishing.



- Walter A. Triebel, Avtar Singh (1991), *The 68000 and 68020 Microprocessors*, Prentice Hall.

Computing Systems

- Rapid pace of information technology is due to introduction of new microprocessors
- Most of us think of desktop computers
  - PC
  - Laptop
  - Mainframe
  - Server



- Maybe at most handheld computer (PDA)
- In this course, we will look at another type of computing system which is far more common that you ever imagined ☺

Computer Classifications

- Classification of computers:
  - Servers:
    - Big, expensive, available 24x7 (read “24 by 7” or 24 hours a day, 7 days a week. Mainframes are old servers made by IBM.
  - Desktops:
    - computers on your desk
  - Laptops:
    - computers you carry in your bag
  - PDA (personal digital assistants):
    - computers you carry in your pocket
  - Embedded systems:
    - computers that don’t look like computers!
- An embedded system is a type of computer

Embedded Systems

- Account for >99% of new microprocessors
  - Consumer electronics
  - Vehicle control systems
  - Medical equipment
  - Sensor networks
- Desktop processors (Intel Core, AMD Athlon, PowerPC, etc) combined is only 1%

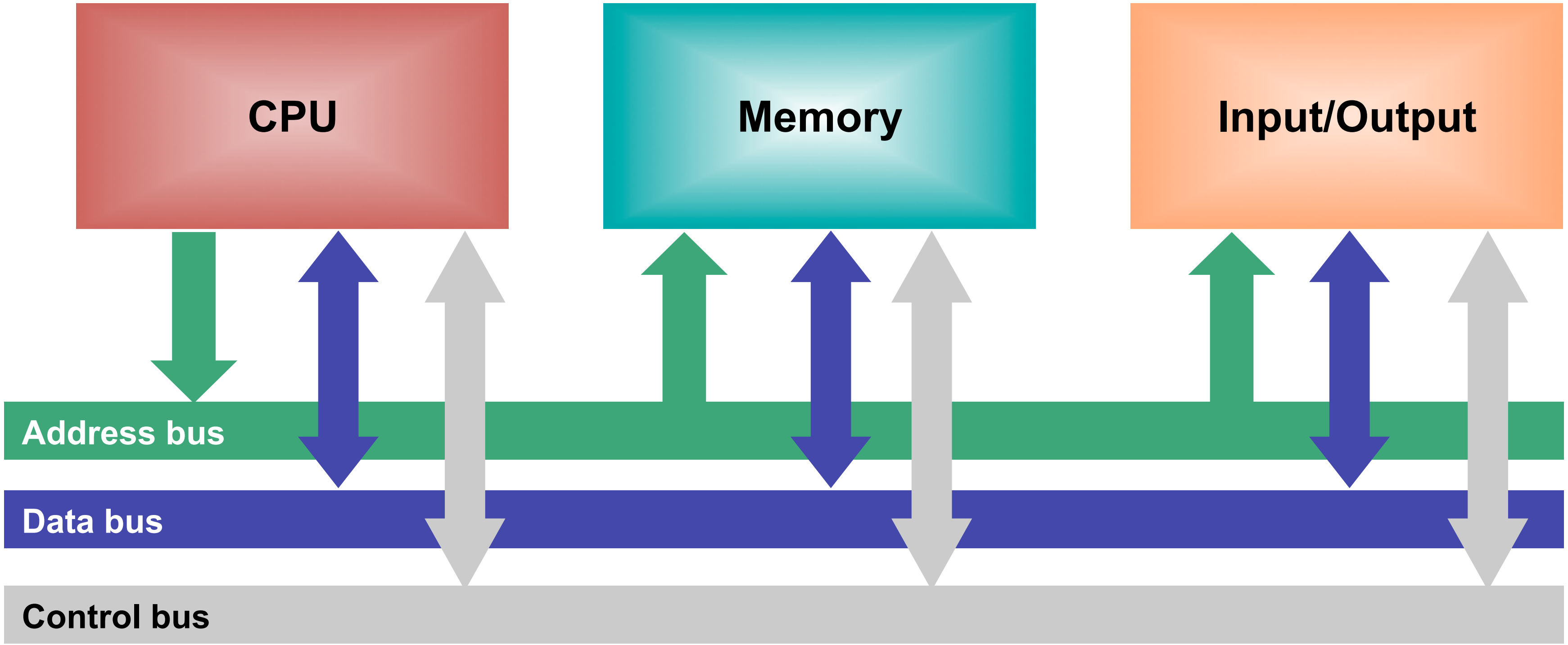
Embedded Systems

- Simple definition: *Computing systems embedded within electronic devices*
- Nearly any computing system other than a desktop computer
- Designed to perform a specific function
- Billions of units produced yearly, versus millions of desktop units
- Take advantage of application characteristics to optimize the design
- As electrical or electronics engineers, you may be required to design an embedded system
  - But you BUY (not design) a general purpose computer

General Purpose vs Embedded Systems

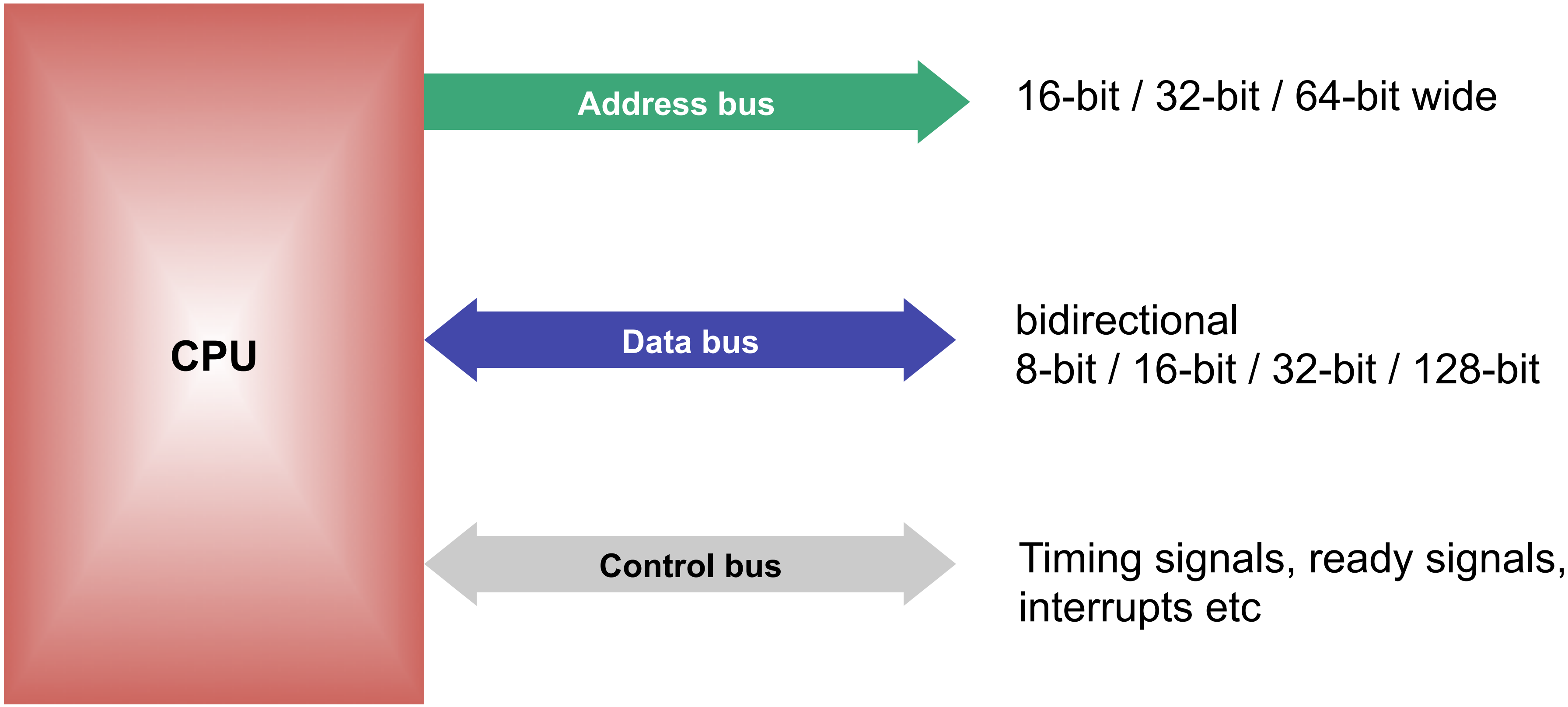
General Purpose	Embedded
Intended to run a fully general set of applications	Runs a few applications often known at design time
End-user programmable	Not end-user programmable
Faster is always better	Operates in fixed run-time constraints, additional performance may not be useful/valuable
Differentiating features: • Speed (need not be fully predictable) • Software compatibility • Cost (eg RM3k vs RM5k per laptop)	Differentiating features: • Power • Cost (eg RM2 vs RM2.50) • Size • Speed (must be predictable)

A Computer System – Simplified View



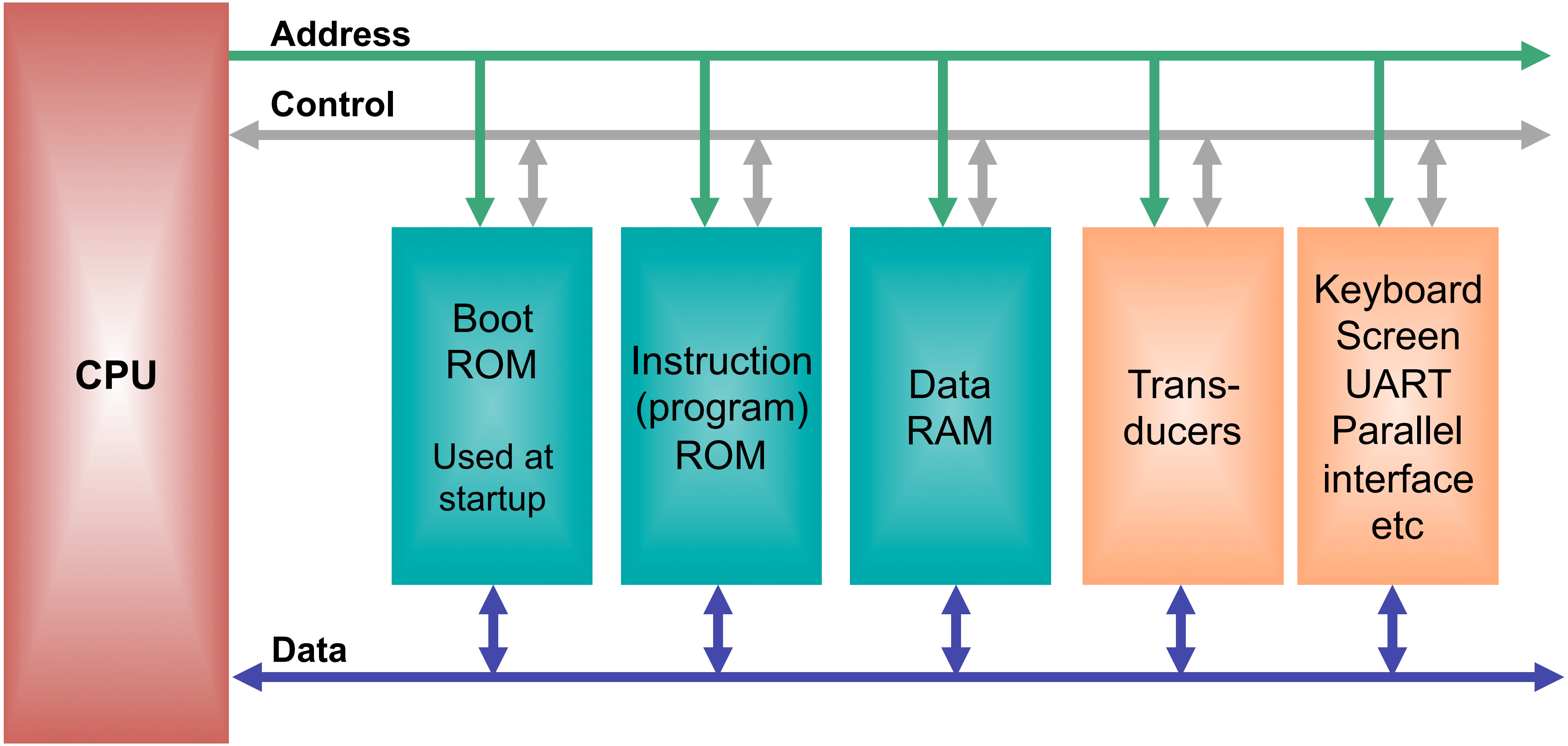
An embedded system also has the same structure but at a smaller size

Microprocessor – Basic concept



Microprocessor, by-itself, completely useless – must have external peripherals to Interact with outside world

Microprocessor – Basic Concept

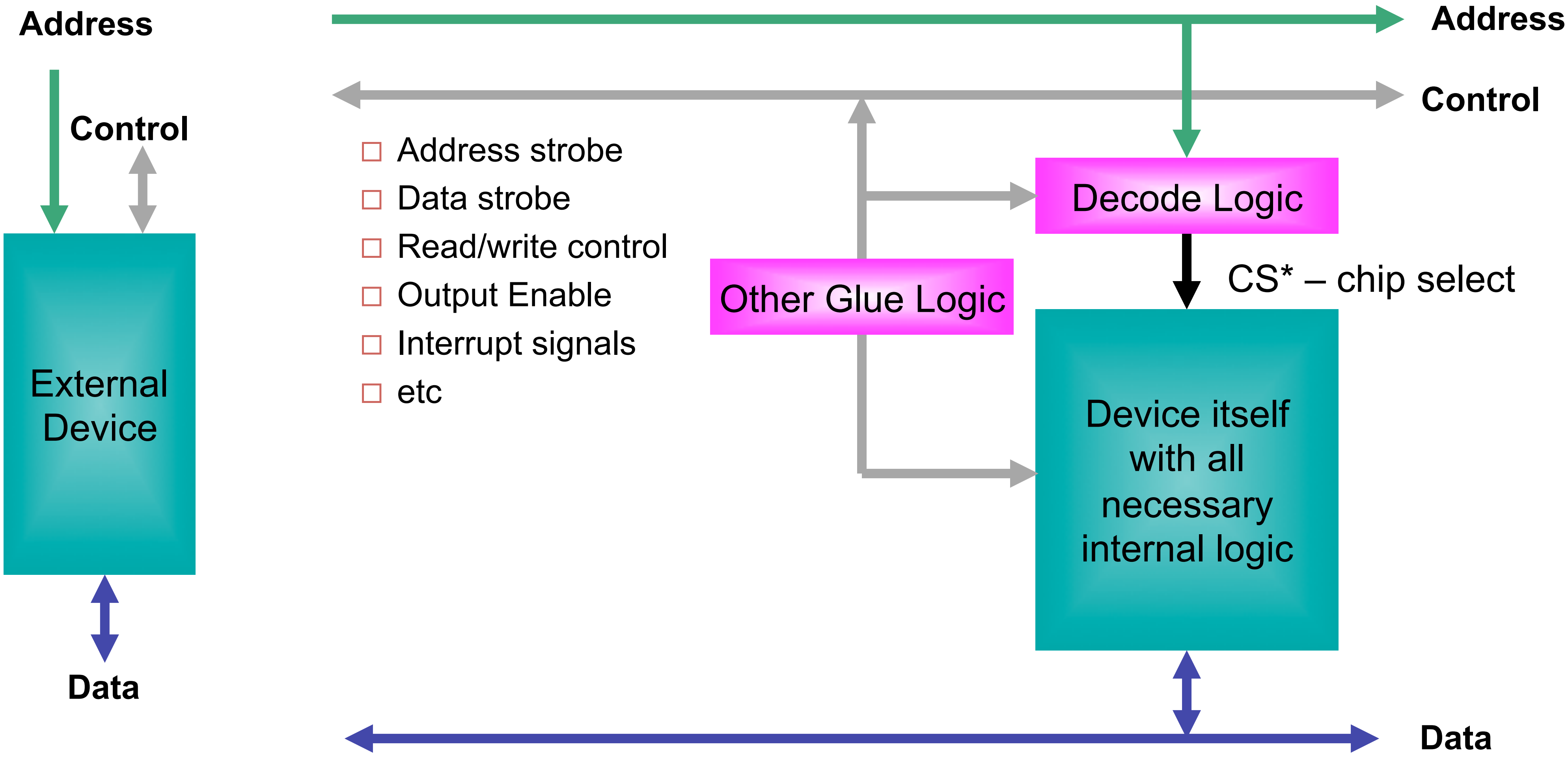


Microprocessor, by-itself, completely useless – must have external peripherals to Interact with outside world

"Glue Logic"

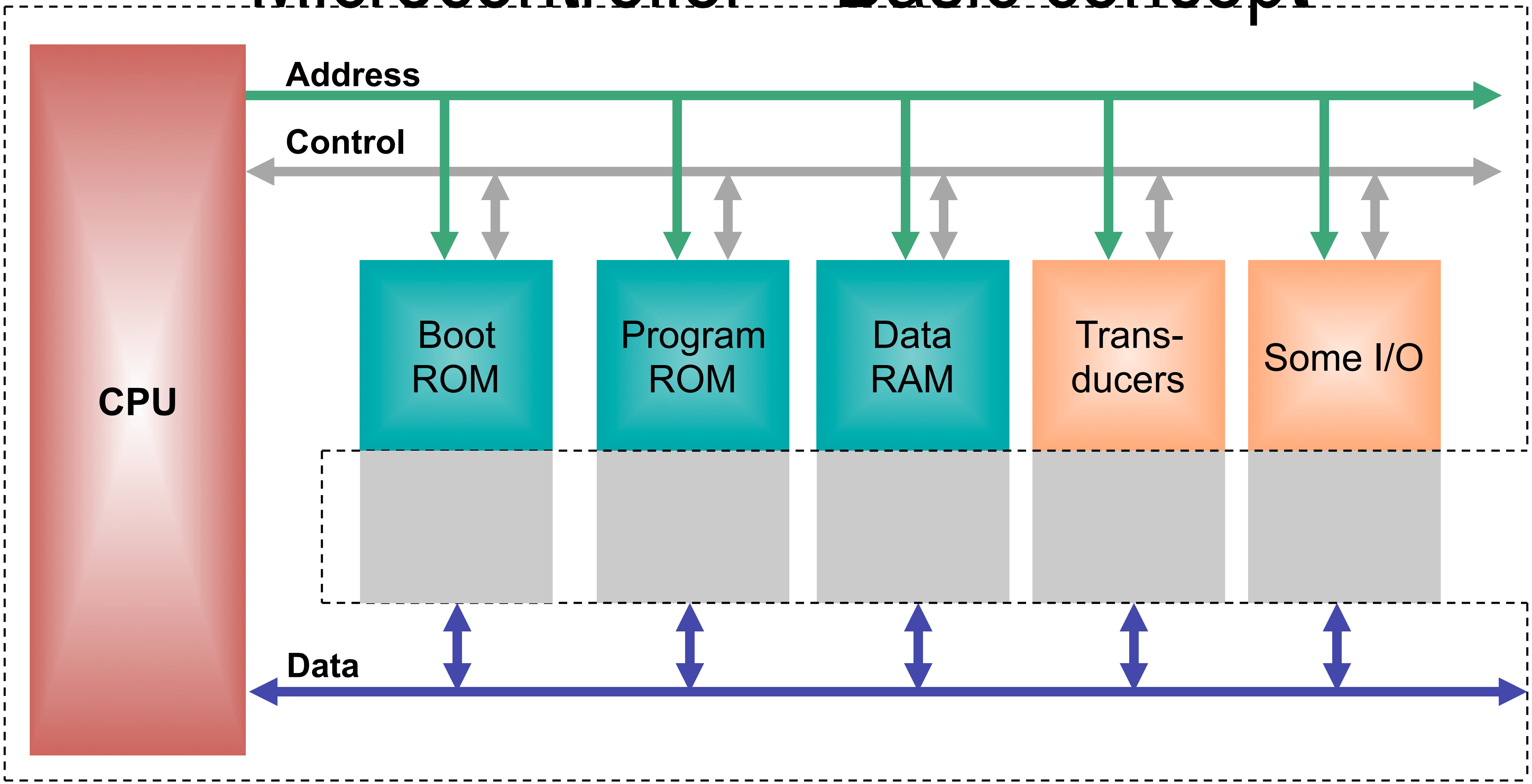
Every external device needs some "glue logic" to interface with the processor.

- Address strobe
- Data strobe
- Read/write control
- Output Enable
- Interrupt signals
- etc



We'll study all the control signals when we study microprocessor hardware.

Microcontroller – Basic concept



Microcontroller - put a limited amount of most commonly used resources inside one chip

Microprocessor vs Microcontroller

- Microprocessor:
  - A chip that contains only the processor
  - Need other chips to make a working system
  - More flexible
  - Can have very few I/O or many I/O devices using the same processor chip

- Microcontroller:
  - A chip that contains all the components of a computer – processor, memory and input/output
  - Less flexibility
  - Less component count in system
  - Less powerful

No matter what is the system size, the most important component is still the processor.

Other Processors in Embedded Systems

- Embedded Controllers:
  - More powerful (32 bits) than microcontrollers (8 bits)
  - Normally contains only processor and input/output, memory is external
- Digital Signal Processors:
  - Embedded processors optimized for digital signal processing
  - Commonly found in handphones, modems, communications systems
- Graphics Processors:
  - Very powerful processors found in graphics cards of workstations
- Programmable Logic Controllers:
  - Microprocessor boards usually found in industrial applications

To design a μP System, we must know…

- Fundamentals:
  - What's inside a computer
  - What's inside a processor
- Programming:
  - What happens in the processor when it's running a program
  - What do we need to write a program
  - How to create a program
  - How to run a program
  - How to fix a program error
- Hardware design:
  - Timing diagrams
  - Interfacing with other chips

Software

- Computer software
  - Computer programs are known as software
- Program:
  - Sequence of instructions that perform a task
  - Think of it like playing music
- Instruction:
  - The simplest operation performed by the processor
  - Think of it as a note coming from a musical instrument
- How the computer works:
  - Fetch an instruction from memory
  - Decode the instruction
  - Execute the instruction
  - Repeat

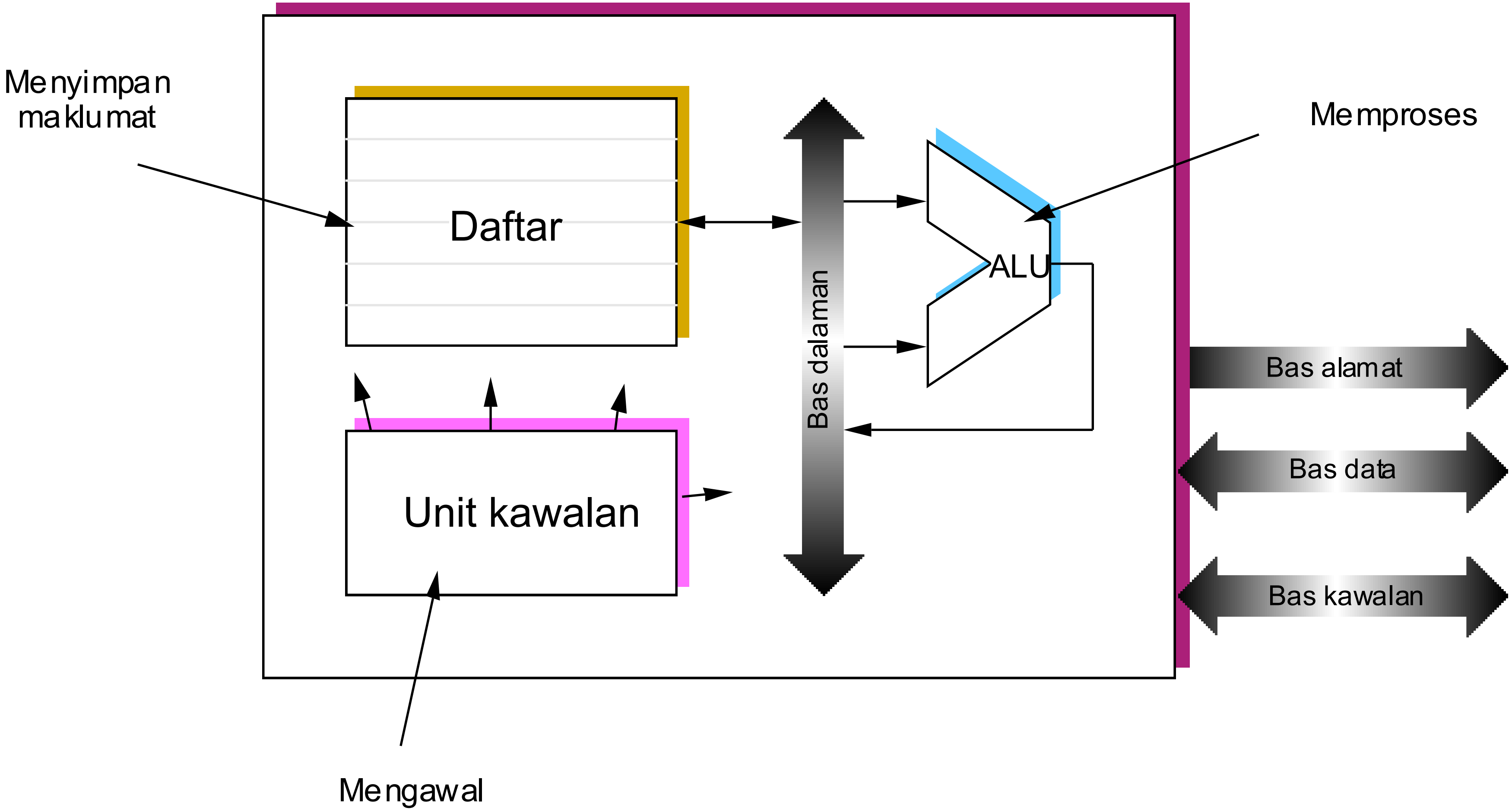
Machine & Assembly Language

- **Machine instruction**
  - A sequence of binary digits which can be executed by the processor, e.g. 0001 1011.
  - Hard to understand for human being
- **Assembly language**
  - An assembly program consists of assembly instructions
  - An assembly instruction is a mnemonic representation of a machine instruction e.g. MUL may stand for "multiply"
  - Assembly programs must be translated into object code before it can be executed -- translated by an assembler.
  - Two types of assemblers: *cross assembler* and *native assembler*.
  - Cross assembler runs on one computer and generates machine instructions that will be executed by another computer that has different instruction set
  - Native assembler runs and generates instructions for the same computer.
  - Drawbacks of assembly programs:
    - Dependent on hardware organisation, difficult to understand long programs, low programmer productivity

High-level language (HLL)

- **High-Level Language**
  - Syntax of a high-level language is similar to English
  - A translator is required to translate the program written in a high-level language into object code -- done by a compiler.
  - There are cross compilers that run on one one computer but translate programs into machine instructions to be executed on a computer with a different instruction set.
  - Main drawback is slower execution speed of the machine code obtained after compiling an HLL program.
  - However, C language has been extensively used in microcontroller programming in industry.

Central Processing Unit (CPU)



Important Registers

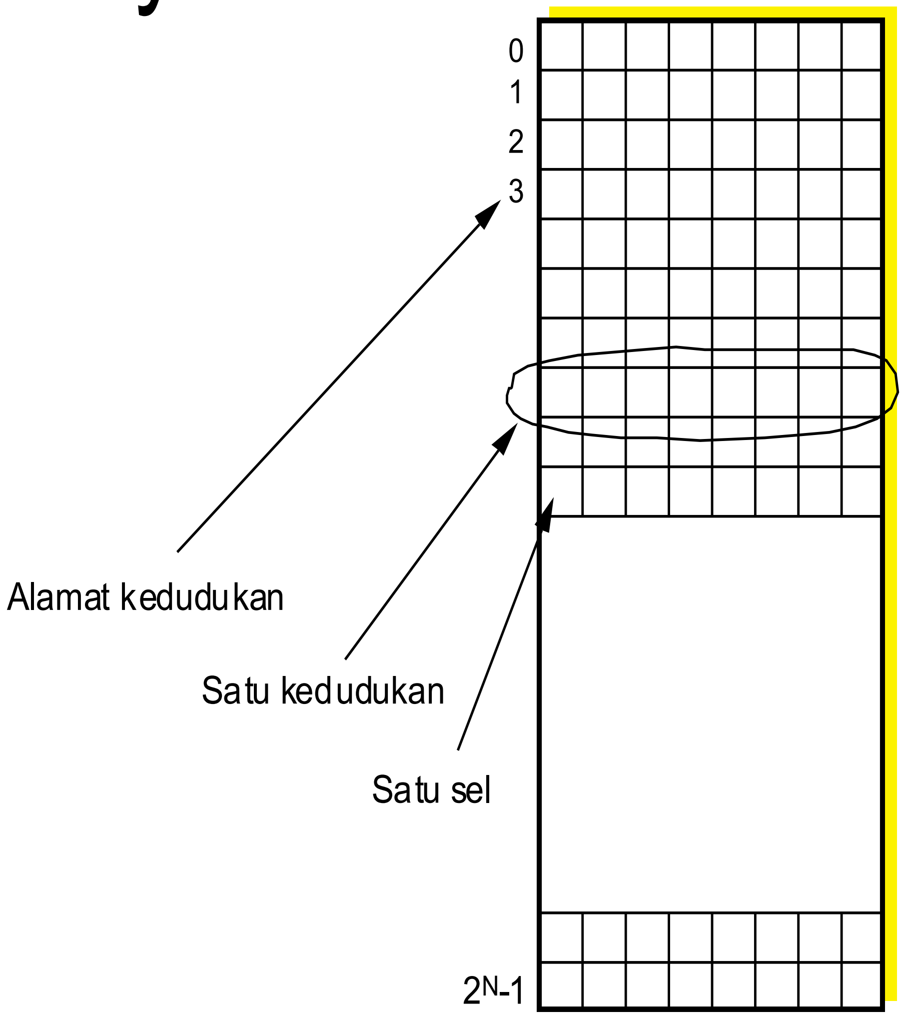
- Program Counter (PC)
  - Keeps track of program execution
  - Address of next instruction to read from memory
  - May have auto-increment feature or use ALU
  - Some manufacturers call this register the Instruction Pointer (IP)
- Instruction Register (IR)
  - Invisible to programmer
  - Contains current instruction
  - Includes ALU operation and address of operand
- Data Registers
  - Stores data. For simple µP, it may be called accumulators.
- Address Registers
  - Stores address of data. For special areas of memory, it may be called index registers, stack pointers or base registers.

The ALU

- Performs arithmetic & logic operations on several bits simultaneously
- The number of bits is a most important factor determining the capabilities of the processor
- Typical sizes:
  - 4 bits (very small microcontroller: remote controllers)
  - 8 bits (microcontrollers: 68HC05, 8051, PIC)
  - 16 bits (low-end microprocessors: Intel 8086)
  - 32 bits (most popular size today: Intel Core, PowerPC, 68000, ARM, MIPS)
  - 64 bits (servers: IBM POWER & PowerPC G5, AMD Opteron, Intel Itanium)

Memory

- Looks like a very long list.
- Each row is called a memory location and has a unique address
- Each location stores the same number of bits, usually multiples of 8 bit (bytes)
- Number of addresses 2^N (where N is an integer).



Memory Devices

- Read-Only Memory
  - Non-volatile memory: contents is retained even without power
  - In embedded systems, used to store application programs and test routines
  - Contents can be set by fixing it during manufacturing or “burning” it using a programming device
  - Common types include MROM, PROM, EPROM and flash memory
  - Erasable types can only be rewritten a fixed number of times
- Random Access Memory
  - Contents lost without power (volatile memory)
  - Used to store temporary data. In embedded system, very little RAM is required. Some systems don’t even have RAM at all!
  - No limit to number of writes the device can handle
  - Fast writes (unlike EPROM/EEPROM)
  - Two major types are SRAM and DRAM

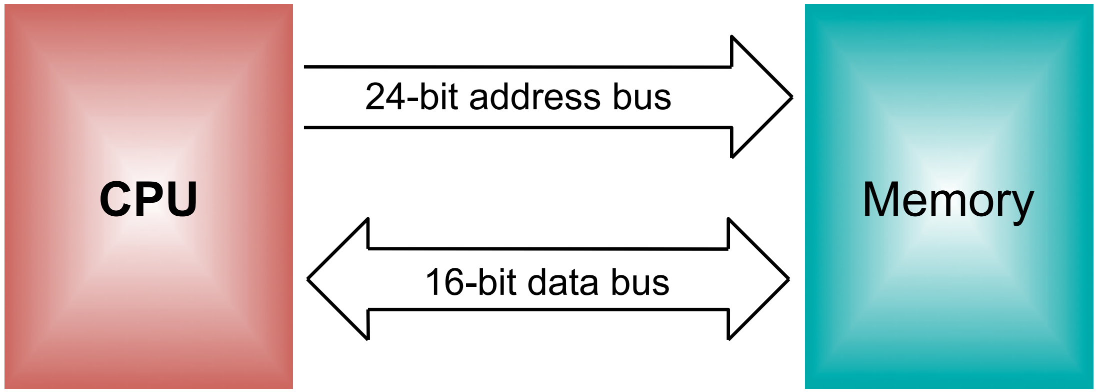
Memory Space and Address Bus

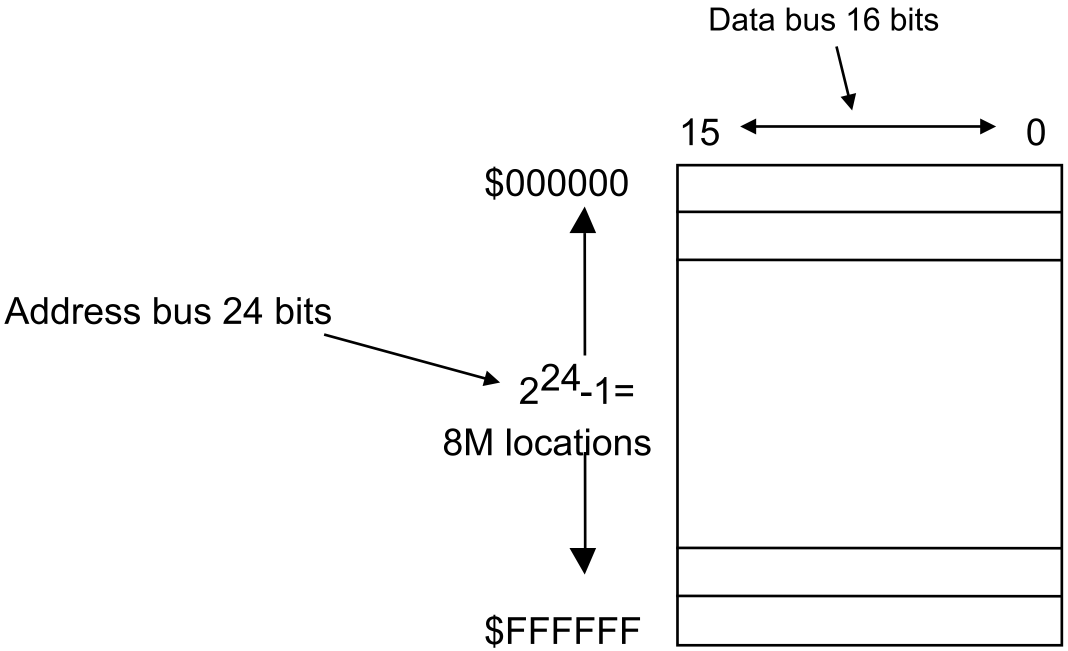
- Smallest transferable amount of data from memory to CPU (and vice versa) is one byte.
- Each byte has a unique location or address.
- The address of each byte is written in hexadecimal (hex).
  - For 68000, the prefix ‘$’ means a hex value.
- The range of addresses accessible by the processor is the **memory space**.
  - Limited by the size of the address bus
- From the programmer’s point of view, 68000 address bus is 24 bits wide.
  - Memory space is 0 to 2^{24}-1 (16777216 or 16 Megabyte)
  - Written in hex as $000000 to $FFFFFF.

Word size and data bus size

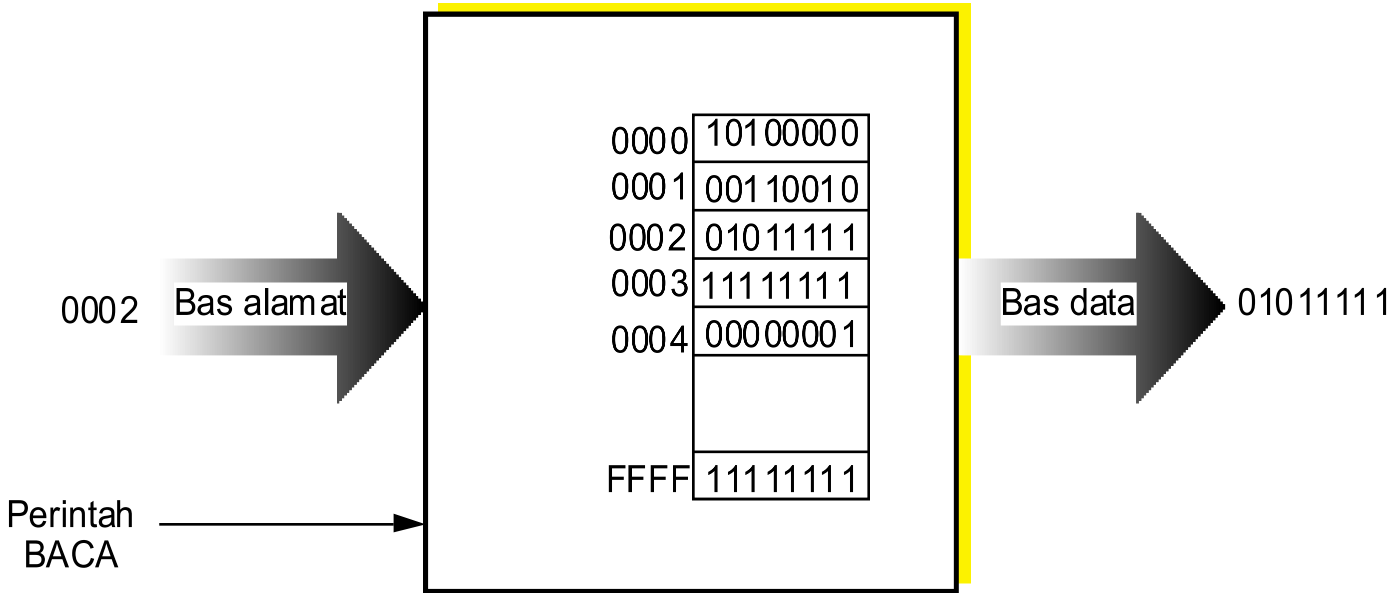
- Width of data bus determines the amount of data transferable in one step
- Original 68000 has a 16 bit data bus
  - Can transfer 1 word or 2 bytes at once
  - A longword requires two transfers
- Current 68HC000 has a selectable bus width of 8 or 16 bits
  - Selecting 8 bit data bus results in cheaper system but lower performance
- The maximum amount of memory for any 68000 system is 16 Mega locations x 1 byte/location = 16 Megabytes
  - Can also be thought of 8 Megawords

Data & Address Buses

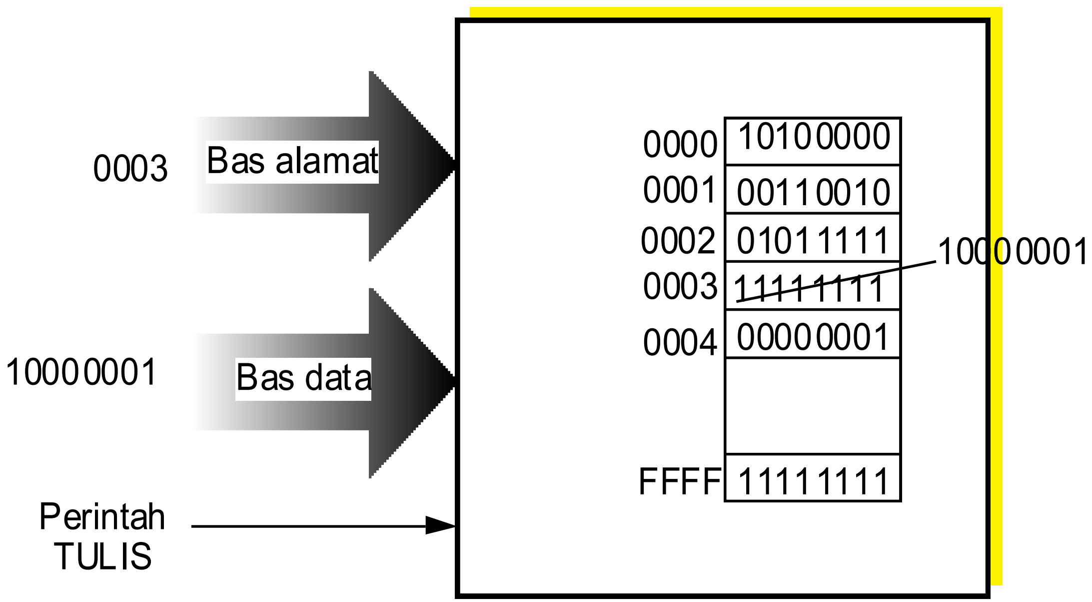




Memory Read Operation

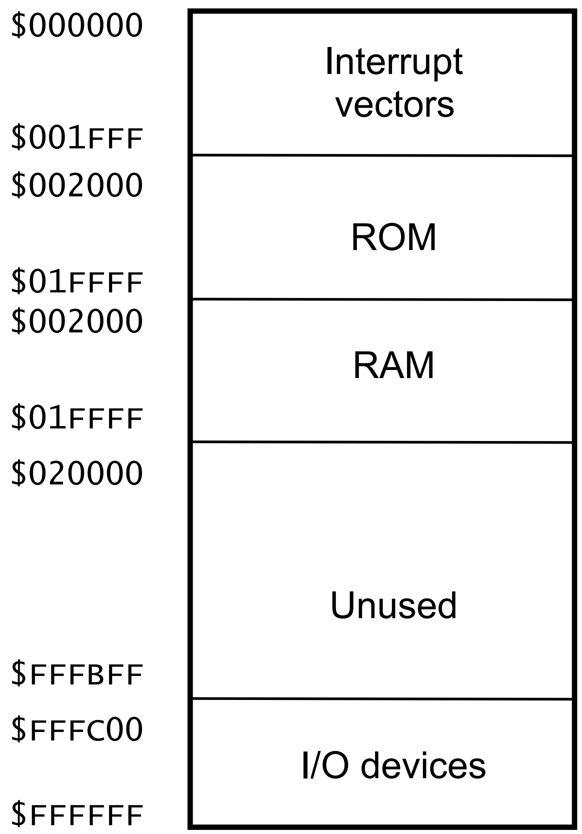


Memory Write Operation



Memory Map

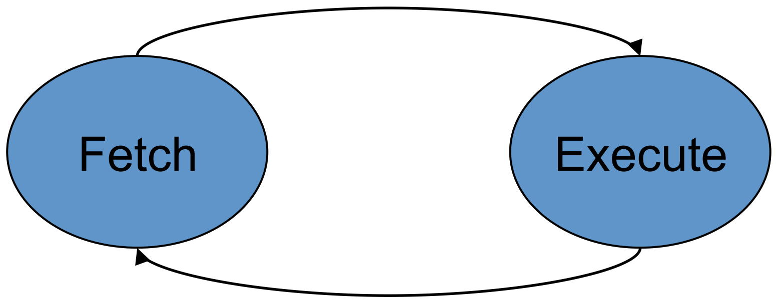
- System memory map summarizes the memory locations available to the programmer
- Must know the following before we can write any program
  - RAM start and end
  - ROM start and end
  - I/O devices
- Very different from writing a program in C where we don't have to know all this



The memory map of a typical system

Fetch-Execute Cycle

- The processor executes instructions one-by-one according to the sequence found in memory
- Everything is controlled by, what else, the **control unit** in the CPU.
- To execute an instruction, the processor must fetch it from memory.
- The complete steps the processor takes to execute one instruction is the **instruction cycle** or the **fetch-execute cycle**



Instruction Cycle Details

- On program start:
  0. Load the program counter (PC) with the address of the first instruction
- Fetch phase:
  1. Read the instruction and put it into the instruction register (IR)
  2. Control unit decodes the instruction; updates the PC for the next instruction
- Execute phase:
  3. Find the data required by the instruction.
  4. Perform the required operation.
  5. Store the results.
  6. Repeat from Step 1.

Instruction Sequencing

- Example – an instruction to add the contents of two locations (A and B) and place result in a third register (C)
- Before you do anything: set PC to point to 1st instruction in the sequence

Program Counter (PC)

12

Instruction Register (IR)

Data Register 0 (D0)

Addr	Instructions	
12	MOVE	A,D0
14	ADD	B,D0
16	MOVE	D0,C
	…	
100	(A) = 4	
102	(B) = 5	
104	(C)	

Instruction Sequencing

Program Counter (PC)

14

Instruction Register (IR)

MOVE A,D0

Data Register 0 (D0)

4

Addr	Instructions	
12	MOVE	A,D0
14	ADD	B,D0
16	MOVE	D0,C
	…	
100	(A) = 4	
102	(B) = 5	
104	(C)	

Instruction Sequencing

Program Counter (PC)

16

Instruction Register (IR)

ADD B,D0

Data Register 0 (D0)

9

Addr	Instructions	
12	MOVE	A,D0
14	ADD	B,D0
16	MOVE	D0,C
	…	
100	(A) = 4	
102	(B) = 5	
104	(C)	

Instruction Sequencing

Program Counter (PC)

18

Instruction Register (IR)

MOVE D0,C

Data Register 0 (D0)

9

Addr	Instructions	
12	MOVE	A,D0
14	ADD	B,D0
16	MOVE	D0,C
	…	
100	(A) = 4	
102	(B) = 5	
104	(C) = 9	

Important Processors You Should Know

Year	Company	Device	Significance
1971	Intel	4004	1st μP. A 4-bit device.
1974	Intel	8008	1st 8-bit μP.
	Motorola	6800	1st 8-bit μP from Motorola.
	Texas	TMS 1000	First microcontroller. Can operate without support chips.
1978	Intel	8086	1st 16-bit μP.
1979	Motorola	68000	16/32-bit μP : the data bus is 16 bits externally, but 32-bit internally.
1984	Motorola	68020	Full 32-bit μP derived from 68000. Has modern features such cache memory, floating-point unit & full support for modern operating systems.
1985	Intel	80386	32-bit μP from Intel, basically unchanged until Pentium of today.
1986	ARM	ARM1	32-bit RISC chips designed for low-power.
1993	Apple/ Motorola/ IBM	PowerPC 601	A RISC chip from Motorola derived from IBM POWER. Ended 68k's use as general purpose computing but the family continues to live in embedded systems until today.

Selecting a Microprocessor

- Choose the right one for your application
  - Primary criteria: Cost, Power, Size, Speed
  - Others: package options, integrated peripherals, potential for future growth
- Choose one with good software development support
  - development environment - good compiler and debugger availability
  - evaluation boards
  - in-circuit emulators for those with deep pockets
  - Operating system availability
- Other considerations
  - Code density: affects power consumption, performance and system cost
  - Hardware availability: make sure you can actually purchase the microcontroller before designing it in
  - Prior expertise, licensing, etc

Summary

- Microprocessors and embedded controllers are a ubiquitous part of life today
- Concept of a microprocessor & microcontroller
- Understand how a µP works
- Headhunters report that EEs familiar with µC, µP design are in the highest possible demand
- Web Resources:
  - How Microprocessors Work:
    - http://computer.howstuffworks.com/microprocessor.htm
    - http://www.intel.com/education/mpworks/
    - http://www.cse.psu.edu/~cg471/03f/hw/pj5/how-micro.html
  - Great Microprocessors of the Past and Present:
    - http://www.sasktelwebsite.net/jbayko/cpu.html
  - Great Moments in Microprocessor History:
    - http://www-128.ibm.com/developerworks/library/pa-microhist.html