

Memory, Circuits, and Decoders

Fundamentals of Microcontroller Memory

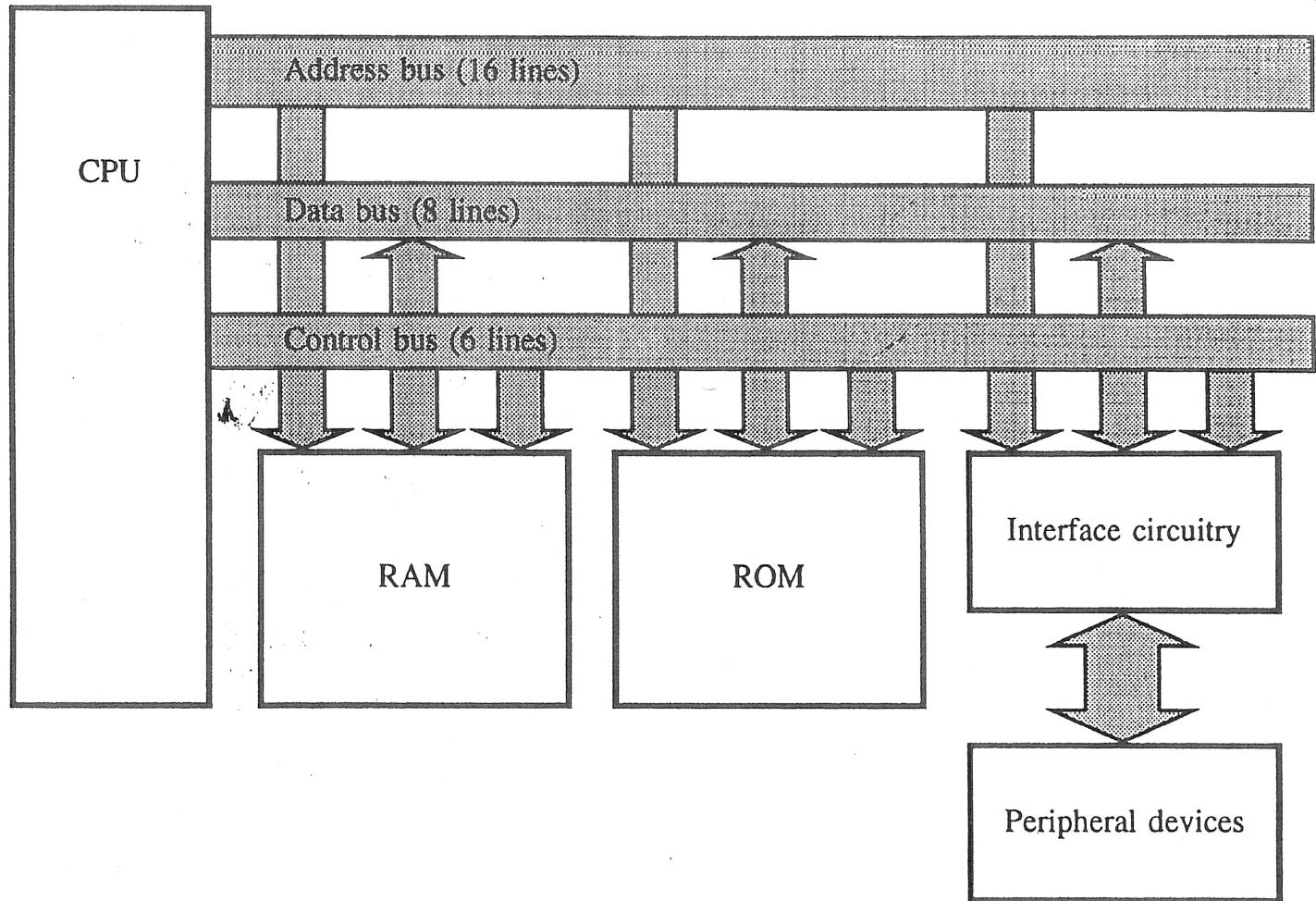


Objectives

- Broad understanding of
 - memory access
 - busses, and
 - decoders
- Understand the role of memory mapping in a computer system

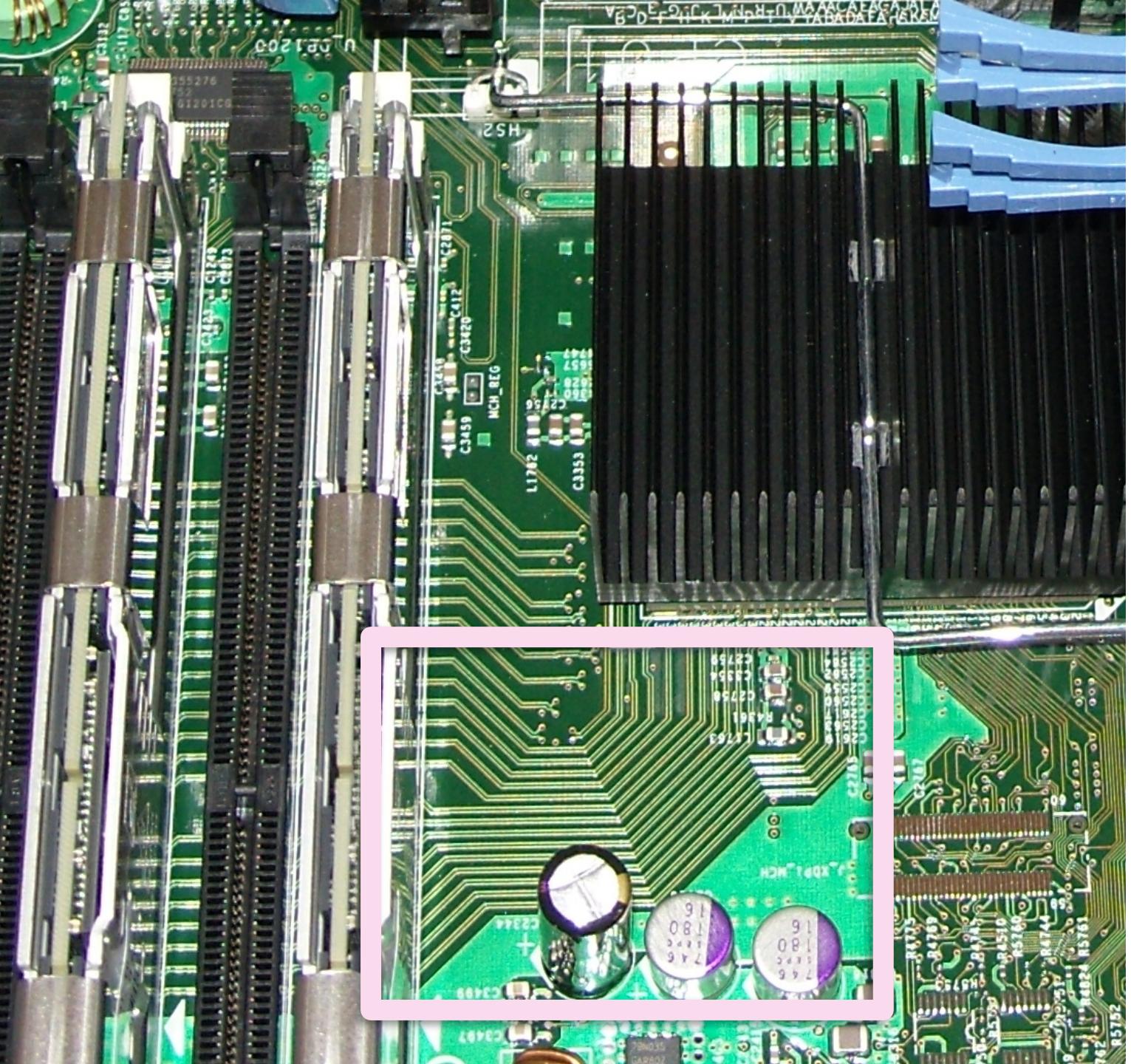
Microcomputer System Overview

- Communication occurs over busses
- A bus is a set of conductive *lines*
 - Address bus: carries the address to be recalled or written
 - Control bus: carries signals such as the clock, read/write lines, etc.
 - Data bus: carries data to and from external systems (bi-directional)



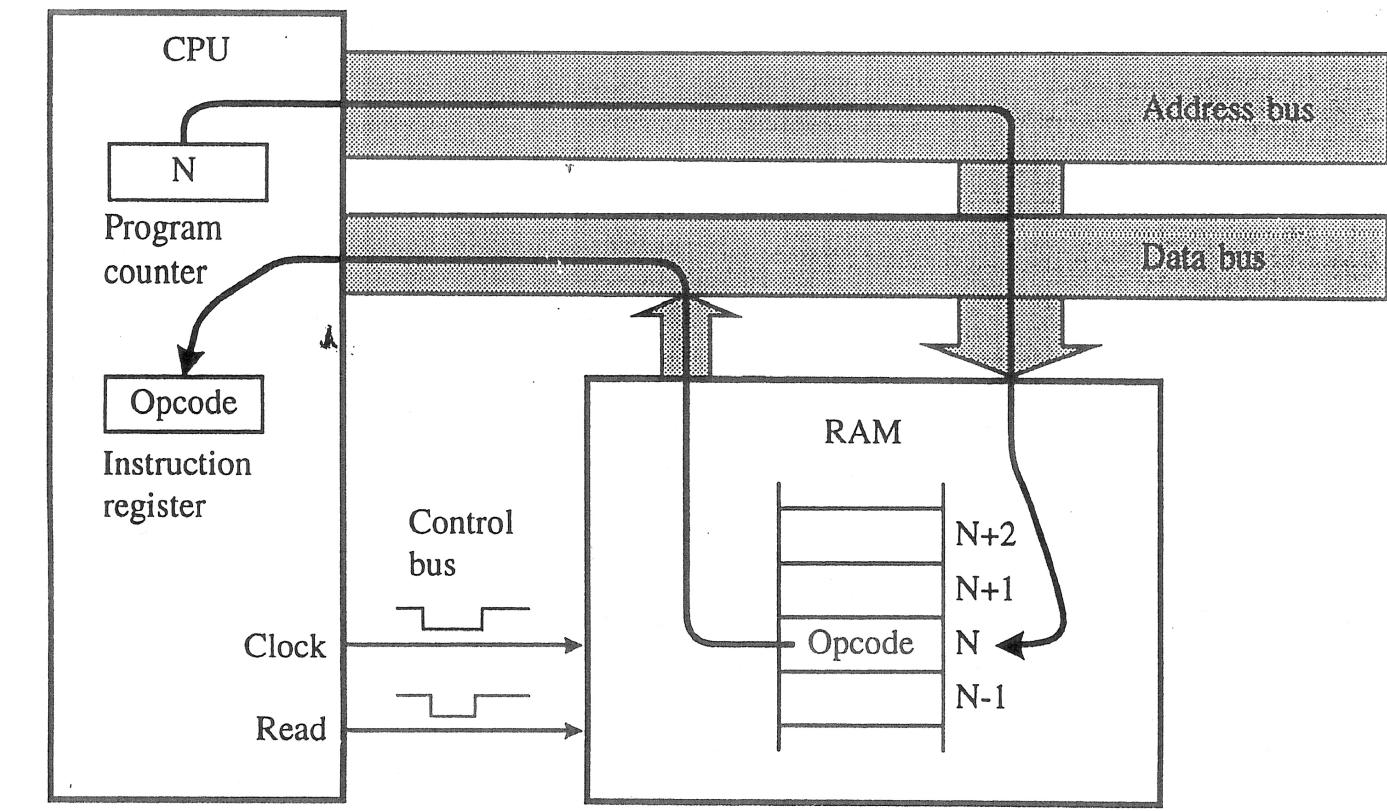
Example Bus Lines

hhedeshian [CC BY-SA (<https://creativecommons.org/licenses/by-sa/3.0>)]



Example Bus Activity: Fetching an Opcode

- Opcodes are instructions executed by the CPU
- 1. CPU places address of next opcode on the address bus
- 2. RAM puts the opcode on the data bus
- 3. CPU reads the opcode into an instruction register and executes it
- Note that the Read and Clock lines are used to tell the RAM what operation is required and to synchronize the process, respectively



Example Opcode from AVR:

ldi: Load a value into a general purpose register

Example: Load register r16 with the value 31

ldi r16, 31

AVR Opcode Example

73. LDI – Load Immediate

73.1. Description

Loads an 8-bit constant directly to register 16 to 31.

Operation:

- (i) $Rd \leftarrow K$

Syntax:

Operands: Program Counter:

- (i) LDI Rd,K

$16 \leq d \leq 31, 0 \leq K \leq 255$

$PC \leftarrow PC + 1$

16-bit Opcode:

1110	KKKK	dddd	KKKK
------	------	------	------

73.2. Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

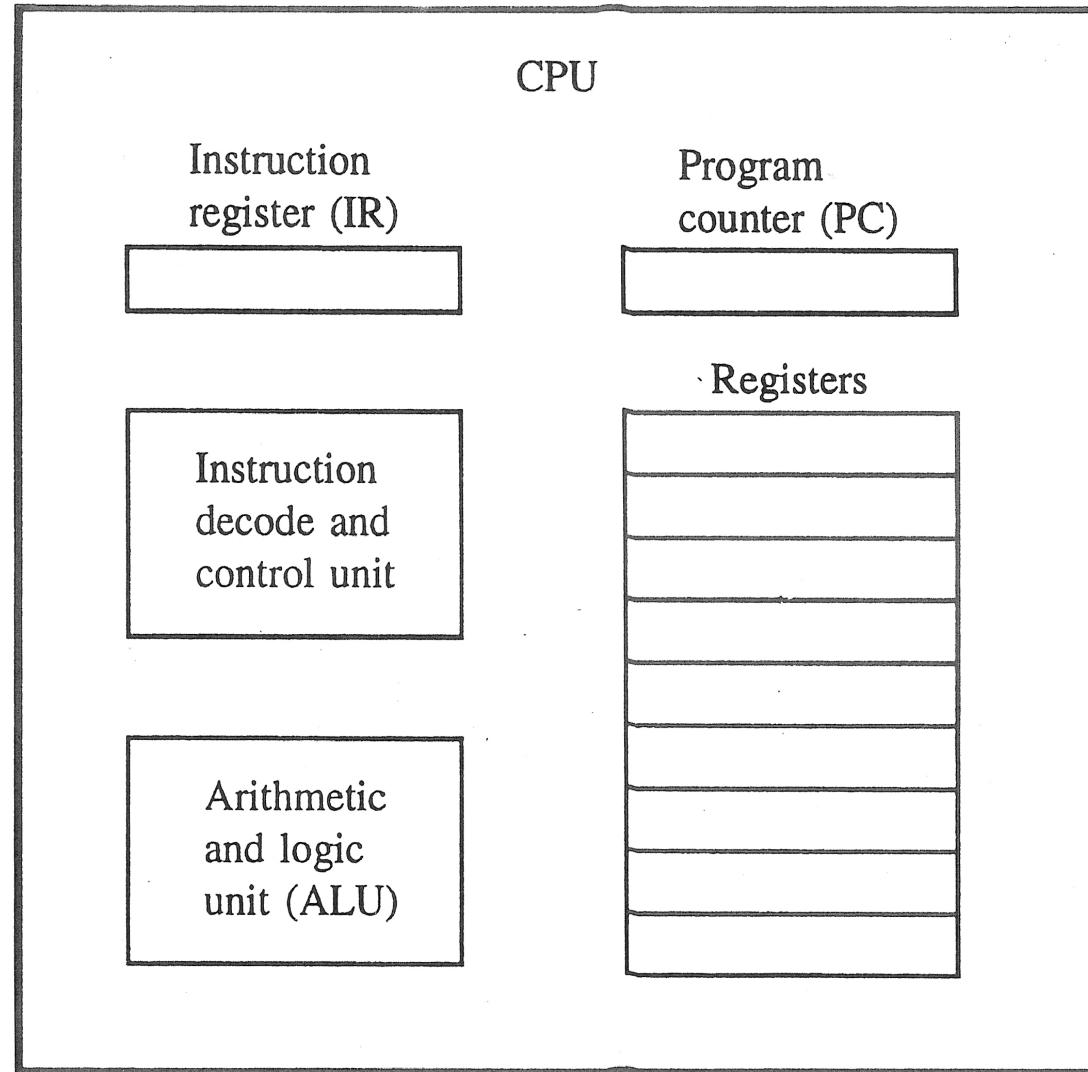
```
clr r31 ; Clear Z high byte
ldi r30,$F0 ; Set Z low byte to $F0
lpm ; Load constant from Program
; memory pointed to by Z
```

Words 1 (2 bytes)

Cycles 1

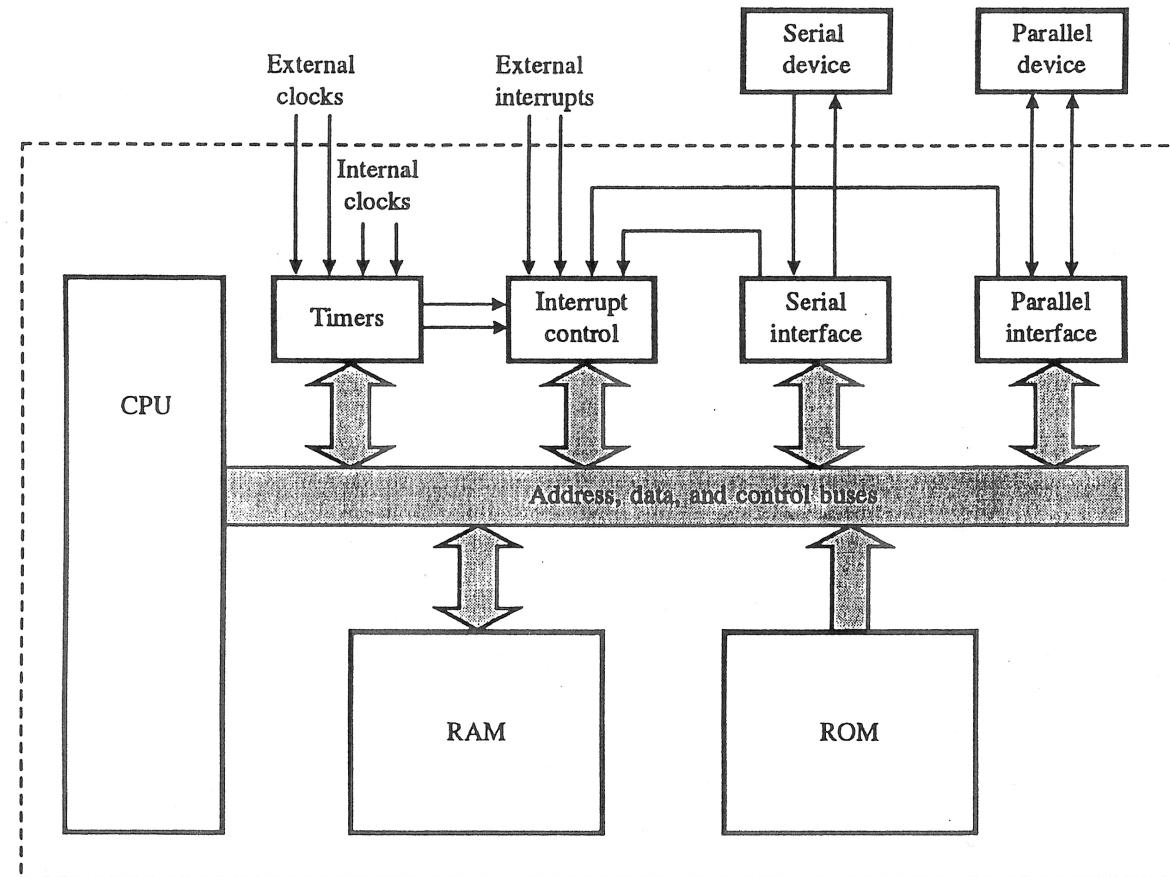
Inside the CPU

- Simplified view of CPU
 - Program counter holds address of next required instruction.
 - Instruction register holds the current instruction just fetched from the program counter address
 - Instr. decode unit analyzes the instruction in the IR to determine the type of operation and executes it
 - ALU performs any required math / logic for the instruction
 - Registers can be used for holding data required for operations



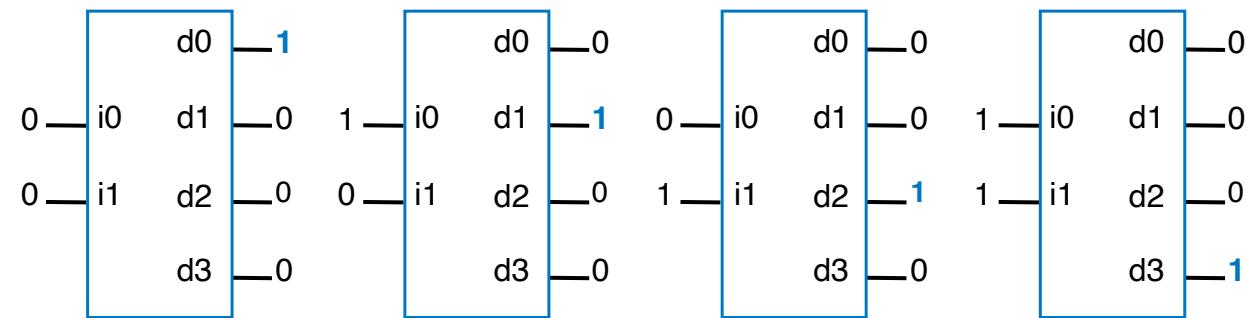
Everything has an Address

- To the CPU, everything is just an address that can be written to or read from
- Addresses are *mapped* to different physical components
- Writing to a serial device, for example, means
 - setting the appropriate control lines
 - setting its address on the address bus, and
 - writing data on the data bus



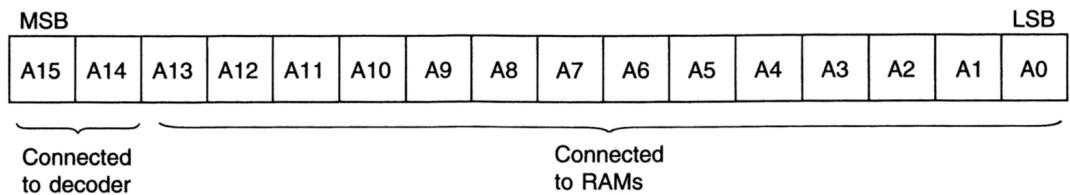
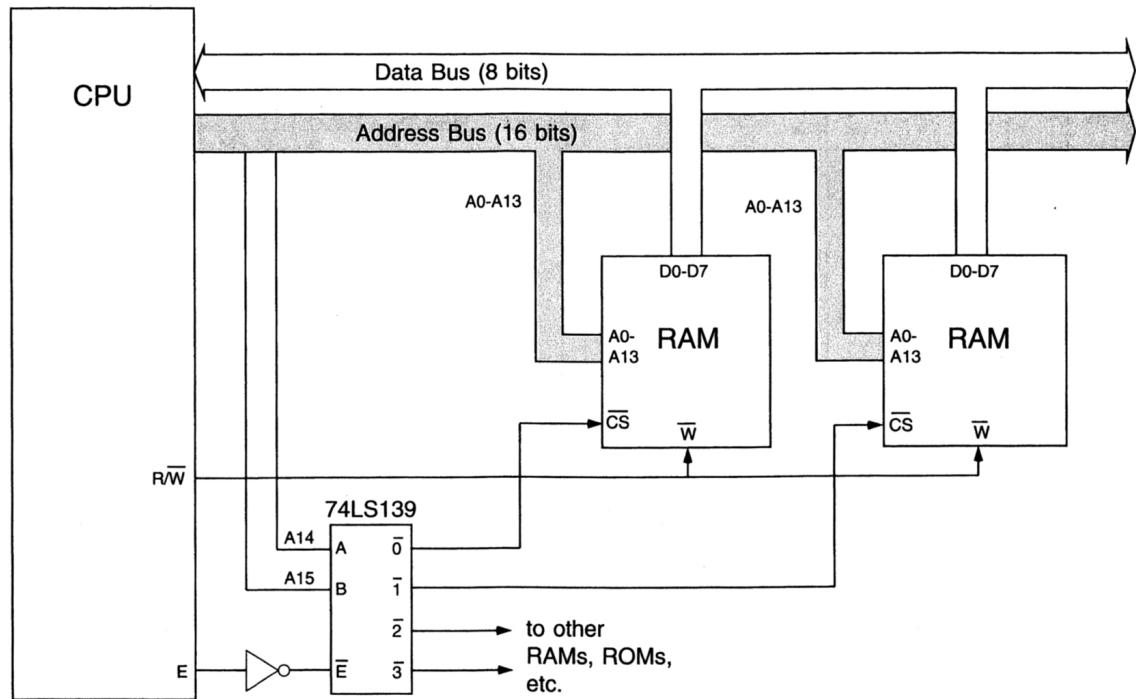
Quick Decoder Review

- Combinational logic building block that converts binary information from n input lines to a maximum of 2^n output lines
- Only one output line is 1 (its position is given by the input binary number)
- Also known as n -to- m line decoders where $m \leq 2^n$



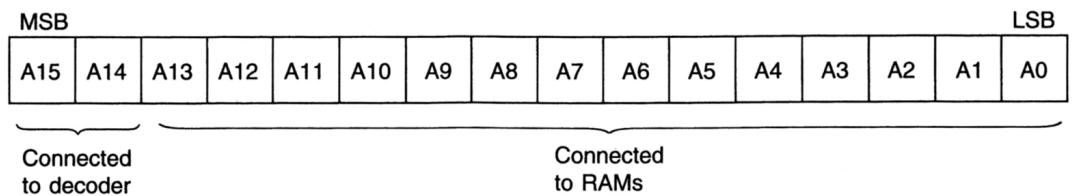
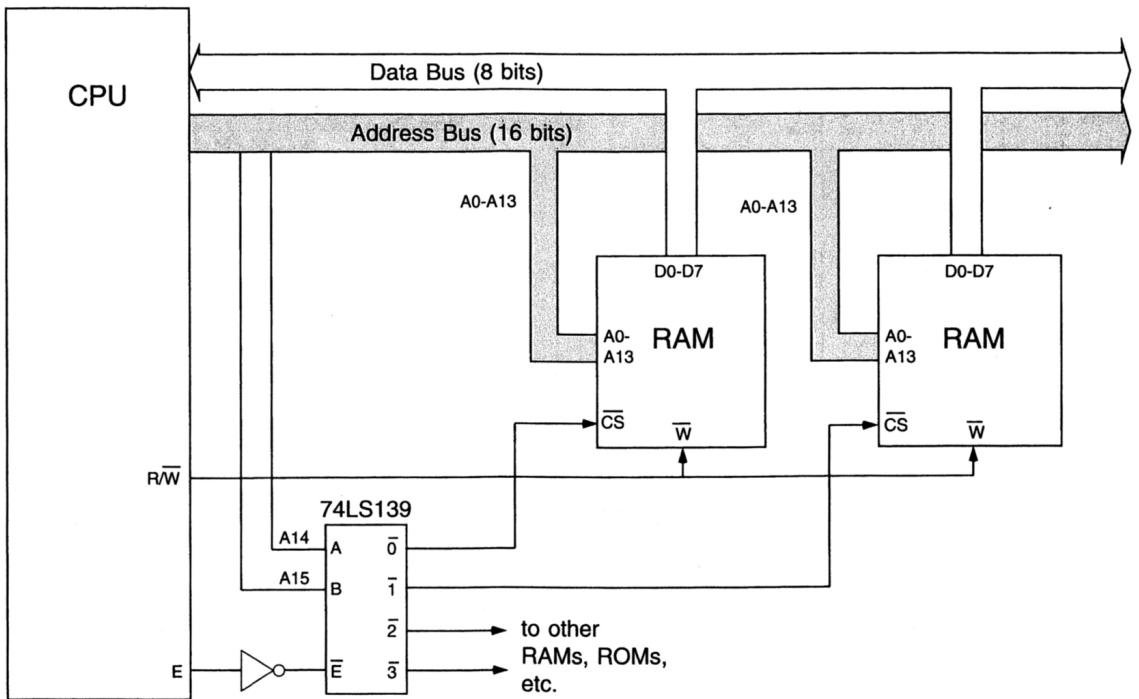
Example of Accessing Physical Memory

- RAM chips are connected to the data and address busses
- The two highest address bits are used to select the memory chips via a decoder
- Each output pin of the decoder can select one RAM chip



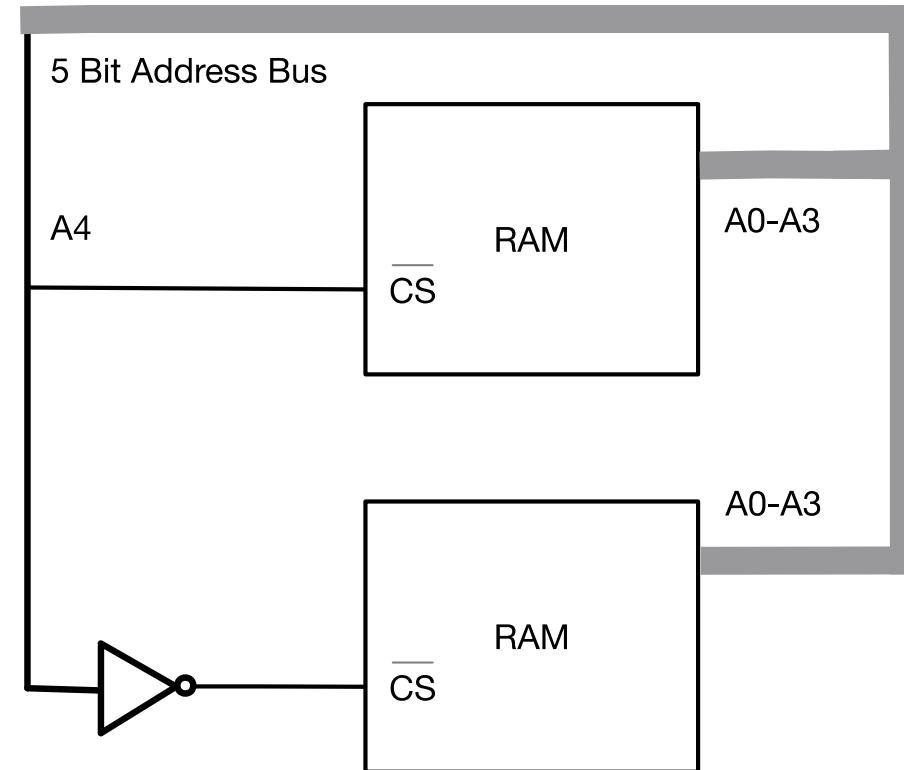
Example of Accessing Physical Memory (2)

- In this configuration, the bit width (8 in this case) remains constant
- Every RAM chip adds more storage to the system
- Each chip is *mapped* to a range of memory
- The range is determined in this case by A14 and A15
- NOTE: The chip selected by A14 and A15 might be a peripheral and not a RAM chip



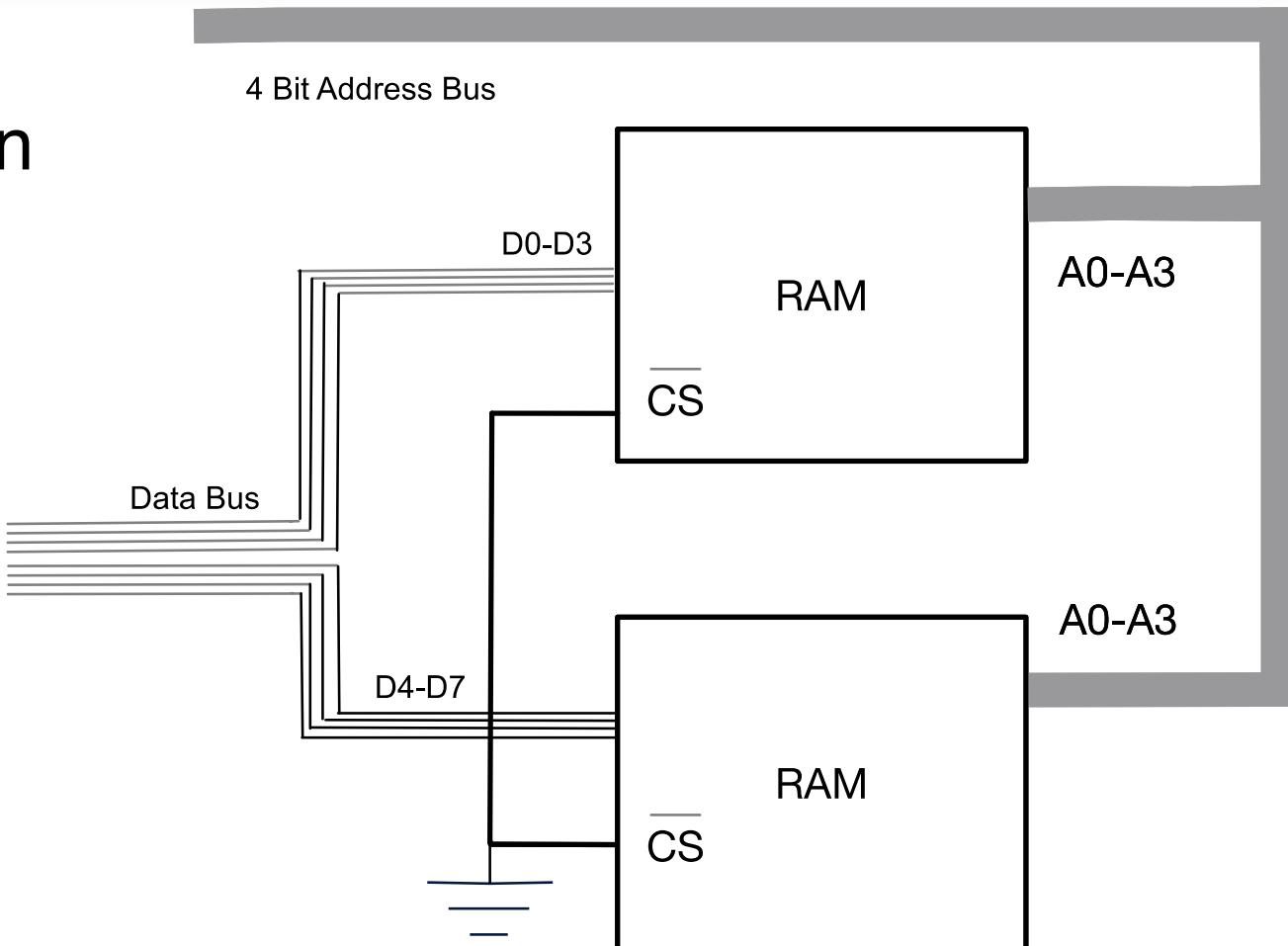
Using Combinational Logic for Chip Selection

- Instead of a decoder, combinational logic circuits can be used to enable / disable ICs
- Example:
 - 01100 retrieves data from address 1100 in the top chip
 - 11100 retrieves data from the bottom chip



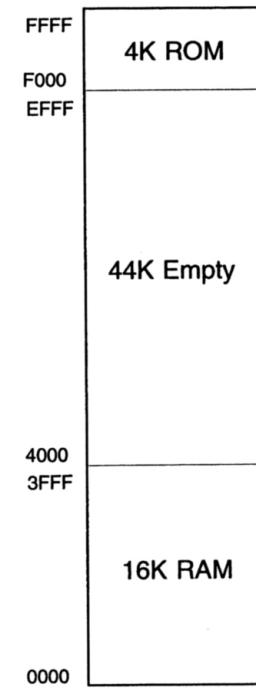
Increasing Word Size

- The same chipset can be configured to increase word size instead of total memory



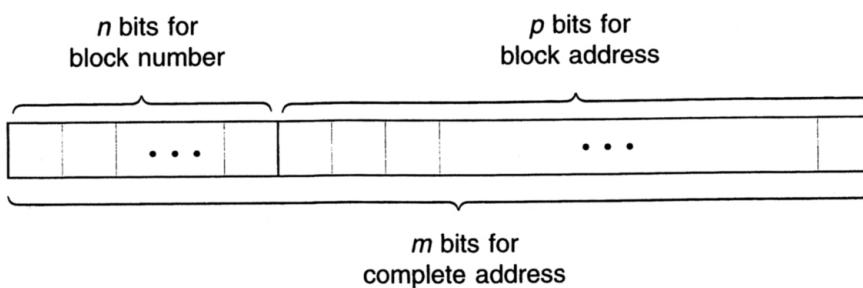
A Memory Map

- A simple memory map for 16 bit system:
 - General purpose RAM 0x0000-to 0x3FFF
 - ROM beginning at 0xF000 and extending to 0xFFFF



Determining Address Bit Allocation

The number of bits needed for block selection depends on the number of divisions required

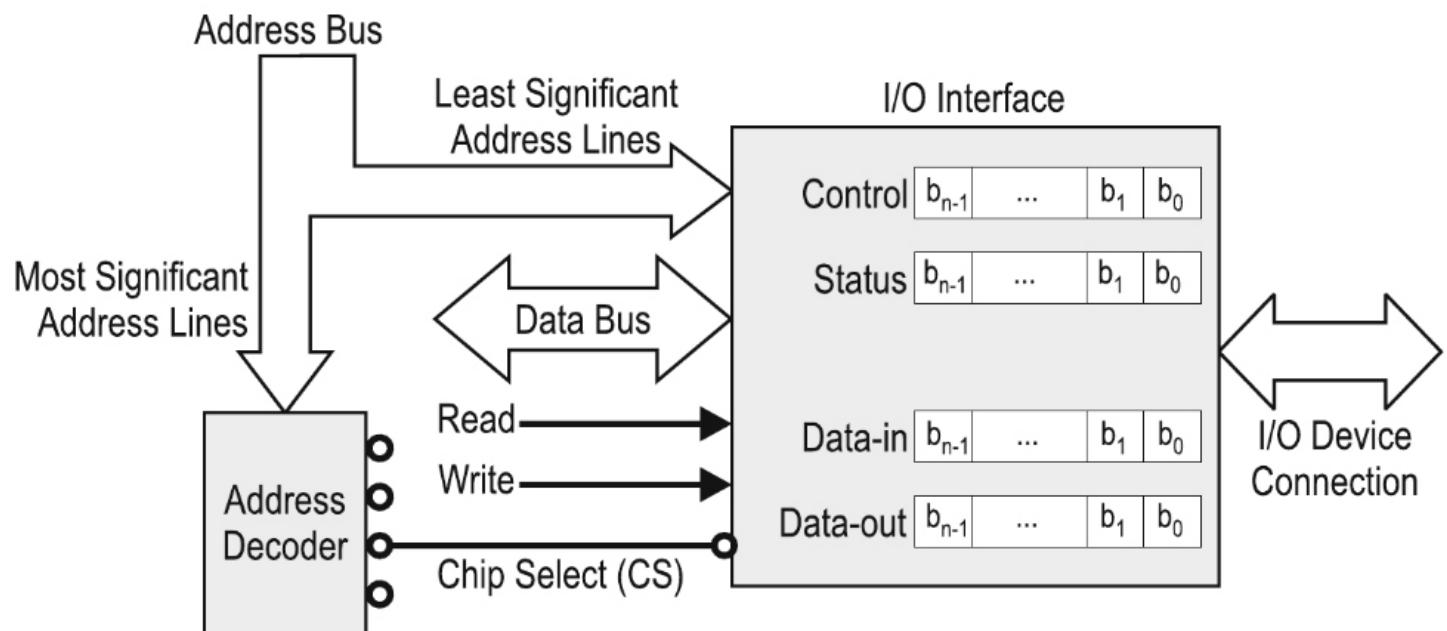


n	p	Number of Blocks (2^n)	Block Size (2^p)
0	16	1	65536
1	15	2	32768
2	14	4	16384
3	13	8	8192
4	12	16	4096
5	11	32	2048
6	10	64	1024
7	9	128	512
8	8	256	256
9	7	512	128
10	6	1024	64
11	5	2048	32
12	4	4096	16
13	3	8192	8
14	2	16384	4
15	1	32768	2
16	0	65536	1

Example for 64K Memory Space

Memory-mapped I/O

- The address decoder uses the MSB address lines to select the appropriate interface
- An interface is mapped to a range of addresses by the decoder
- Internal registers are access by putting the appropriate address on the bus
- Data is read from or written to the registers
- In general, each bit in the Control and Status registers has some specific meaning

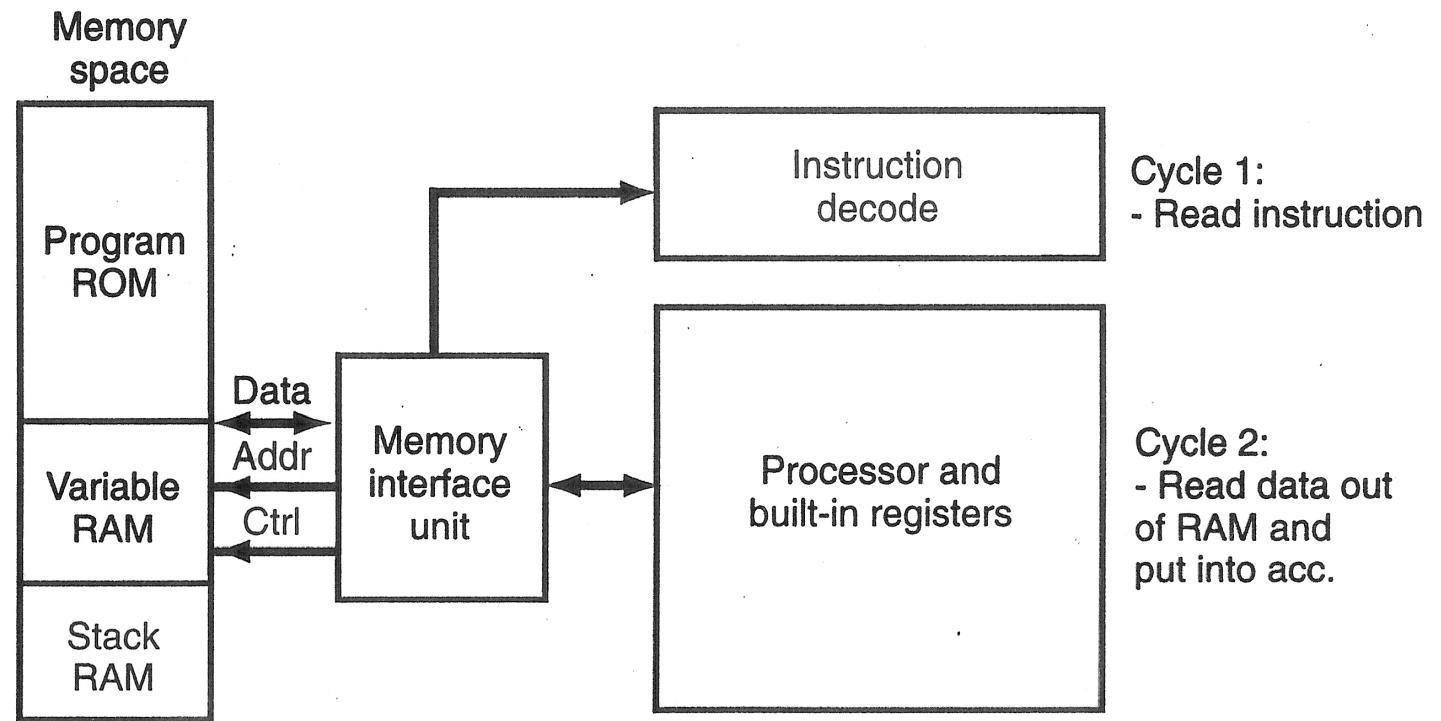


Introduction to Embedded Systems, Jimenez et al

Dividing Program and Data Storage

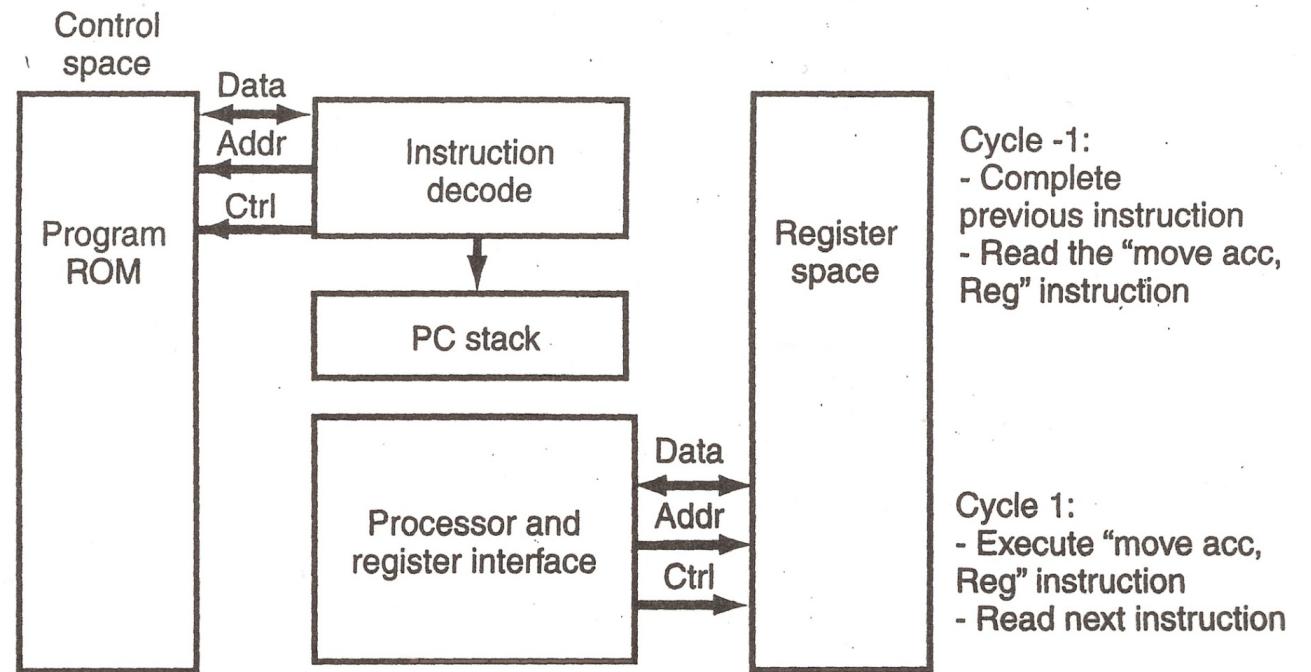
- **Princeton Architecture**

- Data and program memory share the same data and address busses
- The CPU reads the instruction from a location in memory
- Then the CPU reads/writes data to the appropriate memory location



Dividing Program and Data Storage (2)

- **Harvard Architecture**
 - Separate busses for instruction code and data
 - Operations can be completed in one cycle
- Harvard is faster, however the Princeton approach is simpler from a hardware perspective
- NOTE: The AVR chip on the Arduino uses the Harvard architecture



Computer Memory

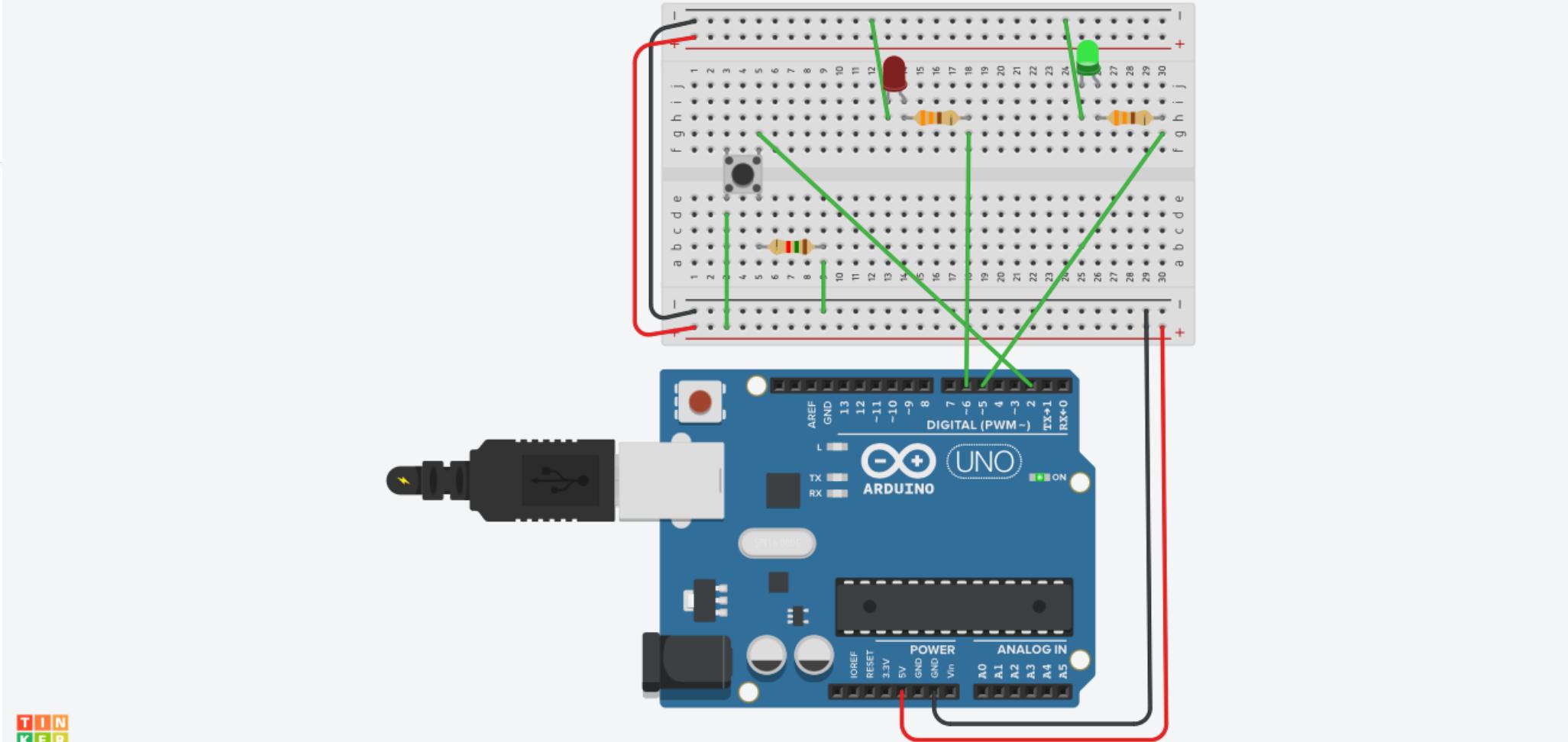
- Types of memory

- Static RAM (SRAM)
- Dynamic RAM (DRAM)
- Flash (NAND/NOR)
- Erasable Programmable Read-Only Memory (EPROM)

The diagram illustrates the classification of computer memory into two main categories: Volatile and Involatile. A pink curly brace groups the first four items (SRAM, DRAM, Flash, EPROM) under the heading 'Volatile - requires power to keep values'. Another pink curly brace groups the last item (EPROM) under the heading 'Involatile – retains values when power is off'.

Volatile - requires power to keep values

Involatile – retains values when power is off



TIN
KER

<https://www.tinkercad.com/things/2L9hSSEInHG-copy-of-lab-1-part2/editel?sharecode=Rsq5D7uMDx-qiL0sIEx2jVKholQmmZFBkEHAifSN1A>

```
// C++ code
//
void setup() {
  Serial.begin(9600);
  Serial.println("Hello from the Serial monitor!");
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(2, INPUT);
}

void loop() {
  char input1 = digitalRead(2);
  if(input1 == LOW ){
    digitalWrite(5,HIGH);
    digitalWrite(6,LOW);
  }
  else if(input1 == HIGH){
    digitalWrite(6,HIGH);
    digitalWrite(5,LOW);
  }
}
```