CSE 315
Microprocessors & Microcontrollers

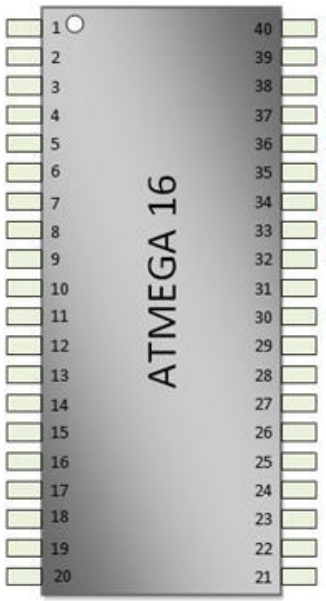
Tanvir Ahmed Khan

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology.

August 24, 2014

Recap

Our Magic Chip



- $V_{cc} \rightarrow 10$
- $GND \rightarrow 11$
- $PortB,\ PB7 \rightarrow 8 - 1 \leftarrow PB0$
- $PortD,\ PD7 \rightarrow 21 - 14 \leftarrow PD0$
- $PortC,\ PC7 \rightarrow 29 - 22 \leftarrow PC0$
- $PortA,\ PA7 \rightarrow 33 - 40 \leftarrow PA0$

Let Us C for Microcontrollers

```
#include <avr/io.h>

int main(void)
{
    //initialization
    while(1)
    {
        //Continuous Processing
    }
    return 0;
}
```

Digital I/O in ATmega16

- 4 8-bit digital IO ports, A, B, C, D
- Each port has 8 data pins
- Every **pin** is bidirectional and can be configured as-
  - input (receiving data in mcu)
  - output (sending data from mcu)

Digital I/O in ATmega16 Contd.

Configuring Pins for Input & Output

- Relevant 8 bit Registers
  - Data Direction Register (DDRx), where, $x = A, B, C, D$
- pin configuration,
  - input $\rightarrow 0$
  - output $\rightarrow 1$
- C Code Example,
  - `DDRA = 0b10101010;`

	o	i	o	i	o	i	o	i
DDRA	1	0	1	0	1	0	1	0
Pin	33	·					·	40

Digital I/O in ATmega16 Contd.

Reading Input Data from Port

- Relevant 8 bit Registers
  - Input Pins Address (PINx), where, $x = A, B, C, D$
- What if some pins are configured as output?
  - we get garbage value
  - so bit mask them before checking
- C Code Example,
  - ```
    unsigned char ch;
    ch = PINA;
    ```
    ```

Digital I/O in ATmega16 Contd.

Writing Output Data to Port

- Relevant 8 bit Registers
  - Data Register (PORTx), where, $x = A, B, C, D$
- What if some pins are configured as input?
- C Code Example,
  - `PORTA = 0b01010101;`

Our Very Simple Counter

Requirements

1. *PA*0 connected with the push button
   - takes pulse input
2. *PORTB* connected with 8 LEDs
   - displays the number of given pulses

Our Very Simple Counter

The C Program

```
 5 #include <avr/io.h>
 6
 7 int main(void)
 8 {
 9     unsigned char ch      = 0;
10     unsigned char counter = 0;
11
12     DDRA  = 0b11111110;
13     DDRB  = 0b11111111;
14
15     PORTB = counter;
16     while(1)
17     {
18         ch = PINA;
19         ch = ch & 1;
20
21         if(ch == 0)
22         {
23             //do nothing
24         }
25         else
26         {
27             counter += 1;
28             PORTB    = counter;
29         }
30     }
31     return 0;
32 }
```

Our Very Simple Counter

The C Program

```
 5 #include <avr/io.h>
 6
 7 int main(void)
 8 {
 9     unsigned char ch      = 0;
10     unsigned char counter = 0;
11
12     DDRA  = 0b11111110;
13     DDRB  = 0b11111111;
14
15     PORTB = counter;
16     while(1)
17     {
18         ch = PINA;
19         ch = ch & 1;
20
21         if(ch == 0)
22         {
23             //do nothing
24         }
25         else
26         {
27             counter += 1;
28             PORTB    = counter;
29         }
30     }
31     return 0;
32 }
```

Can you report any problem?

Our Very Simple Counter

The C Program

```
 5 #include <avr/io.h>
 6
 7 int main(void)
 8 {
 9     unsigned char ch      = 0;
10     unsigned char counter = 0;
11
12     DDRA  = 0b11111110;
13     DDRB  = 0b11111111;
14
15     PORTB = counter;
16     while(1)
17     {
18         ch = PINA;
19         ch = ch & 1;
20
21         if(ch == 0)
22         {
23             //do nothing
24         }
25         else
26         {
27             counter += 1;
28             PORTB    = counter;
29         }
30     }
31     return 0;
32 }
```

We have to add delay

Our Very Simple Counter

The *Delayed* C Program

Version 1

```
 4
 5 #include <avr/io.h>
 6
 7 void delay(void){
 8     int i,j;
 9     for(i=0;i<1000;i++){
10         for(j=0;j<1000;j++){
11             asm volatile("nop");
12         }
13     }
14 }
15
16 int main(void)
17 {
18     unsigned char ch      = 0;
19     unsigned char counter = 0;
20
21     DDRA  = 0b11111110;
22     DDRB  = 0b11111111;
23
24     PORTB = counter;
25     while(1)
26     {
27         ch = PINA;
28         ch = ch & 1;
29
30         if(ch == 0)
31         {
32             //do nothing
33         }
34         else
35         {
36             counter += 1;
37             PORTB    = counter;
38             delay();
39         }
40     }
41     return 0;
42 }
```

Our Very Simple Counter

The *Delayed* C Program

Version 1

```
 5 #include <avr/io.h>
 6
 7 void delay(void){
 8     int i,j;
 9     for(i=0;i<1000;i++){
10         for(j=0;j<1000;j++){
11             asm volatile("nop");
12         }
13     }
14 }
15
16 int main(void)
17 {
18     unsigned char ch      = 0;
19     unsigned char counter = 0;
20
21     DDRA  = 0b11111110;
22     DDRB  = 0b11111111;
23
24     PORTB = counter;
25     while(1)
26     {
27         ch = PINA;
28         ch = ch & 1;
29
30         if(ch == 0)
31         {
32             //do nothing
33         }
34         else
35         {
36             counter += 1;
37             PORTB    = counter;
38             delay();
39         }
40     }
41     return 0;
42 }
```

Version 2

```
 5 #include <avr/io.h>
 6 #include <util/delay.h>
 7
 8 int main(void)
 9 {
10     unsigned char ch      = 0;
11     unsigned char counter = 0;
12
13     DDRA  = 0b11111110;
14     DDRB  = 0b11111111;
15
16     PORTB = counter;
17     while(1)
18     {
19         ch = PINA;
20         ch = ch & 1;
21
22         if(ch == 0)
23         {
24             //do nothing
25         }
26         else
27         {
28             counter += 1;
29             PORTB    = counter;
30             _delay_ms(1000);
31         }
32     }
33     return 0;
34 }
```

Practice Problem

Monitor bit 7 of *PORTB*, if it is 1, configure *PB*4 as input; else change it to output.

Practice Problem

Monitor bit 7 of *PORTB*, if it is 1, configure *PB*4 as input; else change it to output.

```
 5 #include <avr/io.h>
 6
 7 int main(void)
 8 {
 9     unsigned char ch = 0;
10
11     DDRB  = DDRB & 0b01111111;
12
13     while(1)
14     {
15         ch = PINB;
16         ch = ch & 0b10000000;
17
18         if(ch == 0b10000000)
19         {
20             DDRB = DDRB & 0b11101111;
21         }
22         else
23         {
24             DDRB = DDRB | 0b00010000;
25         }
26     }
27     return 0;
28 }
```

Important Notices

Important Notices

- No project in CSE 316, ☺, or ☹?

Important Notices

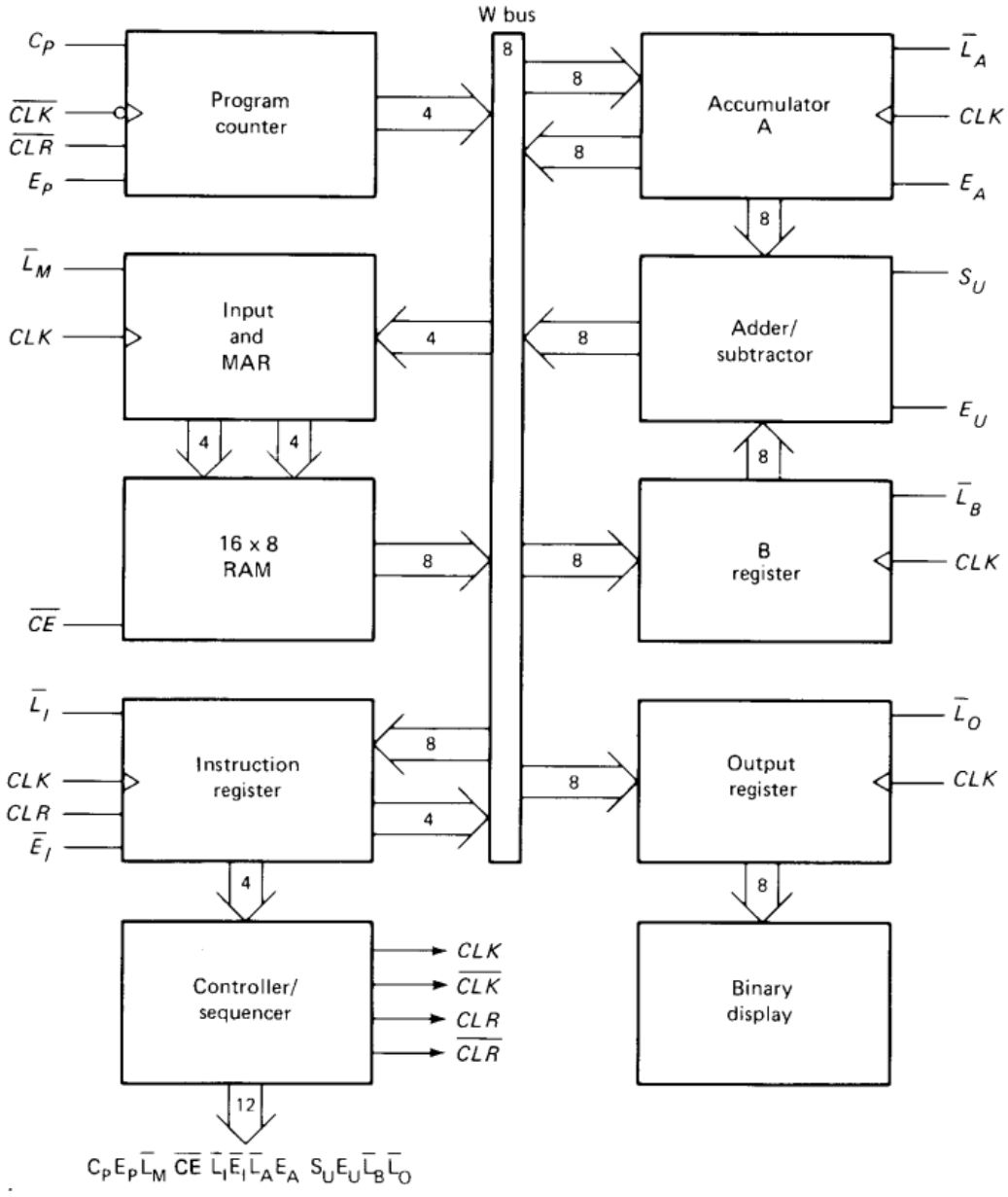
- No project in CSE 316, ☺, or ☹?
- Microcontrollers Resources in IAC

Computer Architecture Revisited

Simple As Possible: SAP Computer

In Detail Explanation
in
Next term

SAP Architecture



Today's Topic

ATmega16 Architecture

ATmega16 Architecture

Overview

ATmega16 Architecture

Overview

- Reduced Instruction Set Computer

ATmega16 Architecture

Overview

- Reduced Instruction Set Computer
- Harvard Architecture

ATmega16 Architecture

Overview

- Reduced Instruction Set Computer
- Harvard Architecture

for fast and efficient program execution

Reduced Instruction Set Computer

Reduced Instruction Set Computer

- register-based architecture
  - 32 8-bit registers coupled with ALU within CPU

Reduced Instruction Set Computer

- register-based architecture
  - 32 8-bit registers coupled with ALU within CPU
- instruction set based on RISC concept
  - 131 RISC-type (mostly single clock cycle) instructions

Harvard Architecture

Harvard Architecture

- separate, dedicated memories and buses for instruction and data

Harvard Architecture

- separate, dedicated memories and buses for instruction and data
- 3 main memory sections

Harvard Architecture

- separate, dedicated memories and buses for instruction and data
- 3 main memory sections
  - In-System Re-programmable nonvolatile Flash Memory, 16 KB

Harvard Architecture

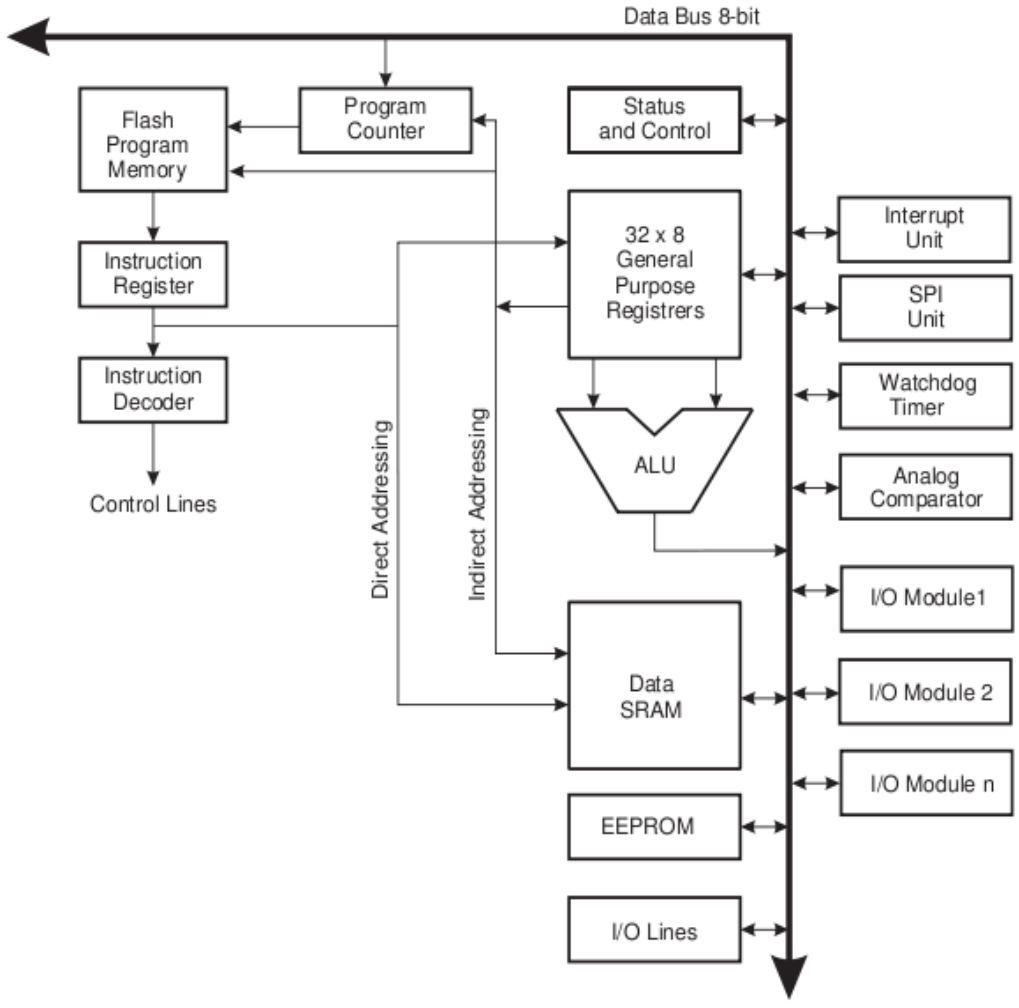
- separate, dedicated memories and buses for instruction and data
- 3 main memory sections
  - In-System Re-programmable nonvolatile Flash Memory, 16 KB
  - volatile SRAM to feature stack and data memory, 1120 B

Harvard Architecture

- separate, dedicated memories and buses for instruction and data
- 3 main memory sections
  - In-System Re-programmable nonvolatile Flash Memory, 16 KB
  - volatile SRAM to feature stack and data memory, 1120 B
  - nonvolatile EEPROM, 512 B

Atmega16 Architecture

Block Diagram



Reference

- Atmel AVR Microcontroller Primer: Programming and Interfacing, *Chapter 1*
  - Steven F. Barrett
  - Daniel J. Pack