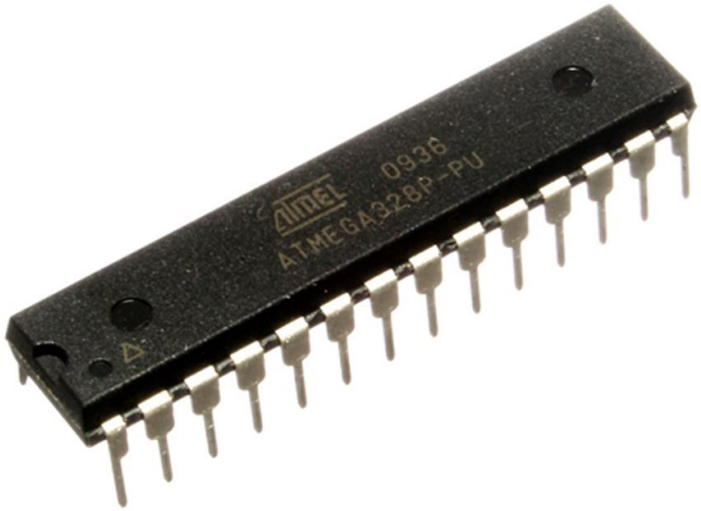
Machine Architecture - Lecture 7



Ioannis Ivrissimtzis

ioannis.ivrissimtzis@durham.ac.uk

ATmega328P: registers, memory map, I/O

Slide material acknowledgements: Magnus Bordewich

In this lecture …

This lecture will be heavy in terminology.

There will be a lot of acronyms and technical details.

You do not have to remember them.

Ideally, another skill that you will acquire from this part of the course will be the ability to engage with technical documents, such as the megaAVR® Data Sheet and the AVR Instruction Set Manual uploaded on Ultra.

Overview

The ATmega328P microcontroller:

- main technical characteristics
- registers
- memory map
- I/O

Microcontroller

Microprocessor vs microcontroller

Microcontrollers are integrated devices (computers in a chip) consisting of

- a microprocessor (including ALU, PC, and registers)
- memory (Flash, SRAM)
- input / output

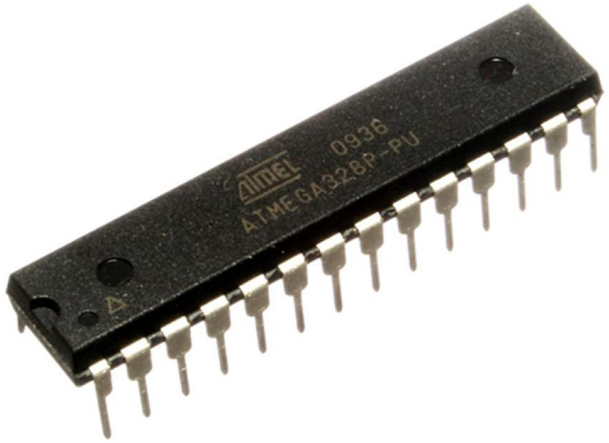
They are used in embedded systems, that is, special purpose computer systems designed to perform dedicated functions, often in real-time.

The program is usually stored in Flash memory, and it might never change during the lifetime of the microcontroller.

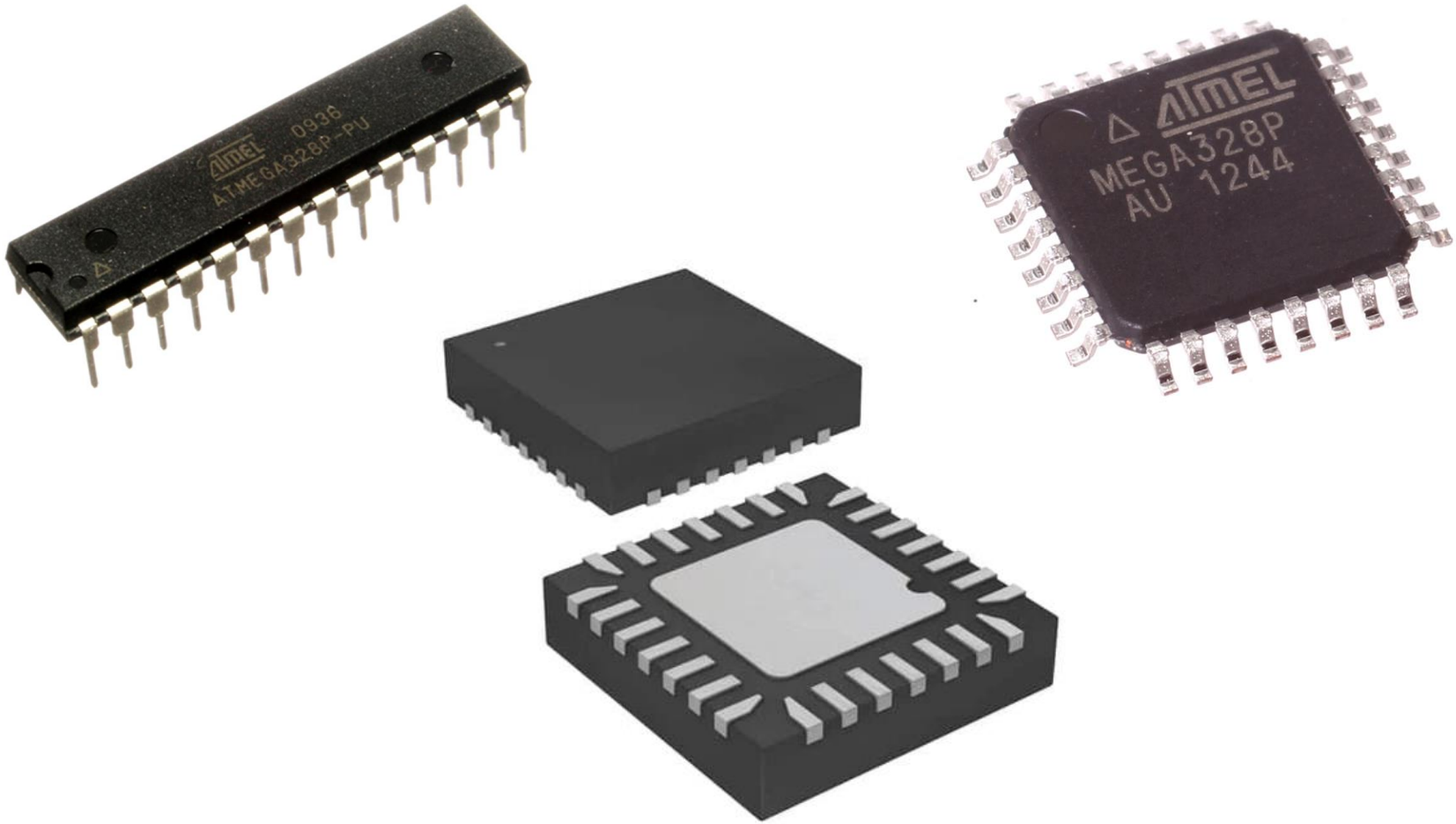
Main characteristics

Arduino UNO uses the ATmega328P microcontroller:

- Harvard architecture
- 8-bit processor (8-bit registers, 8-bit bus)
- 16-bit words (but some instructions are 32-bit)



Various packages



Overview

The ATmega328P microcontroller:

- main technical characteristics
- registers
- memory map
- I/O

Registers

32 general purpose 8-bit registers.

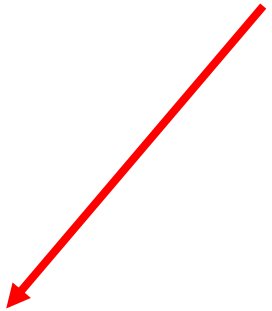
They are also mapped (copied) on the SRAM memory.

Other SRAM memory locations are also used as “registers”, that is, they are designated as registers.

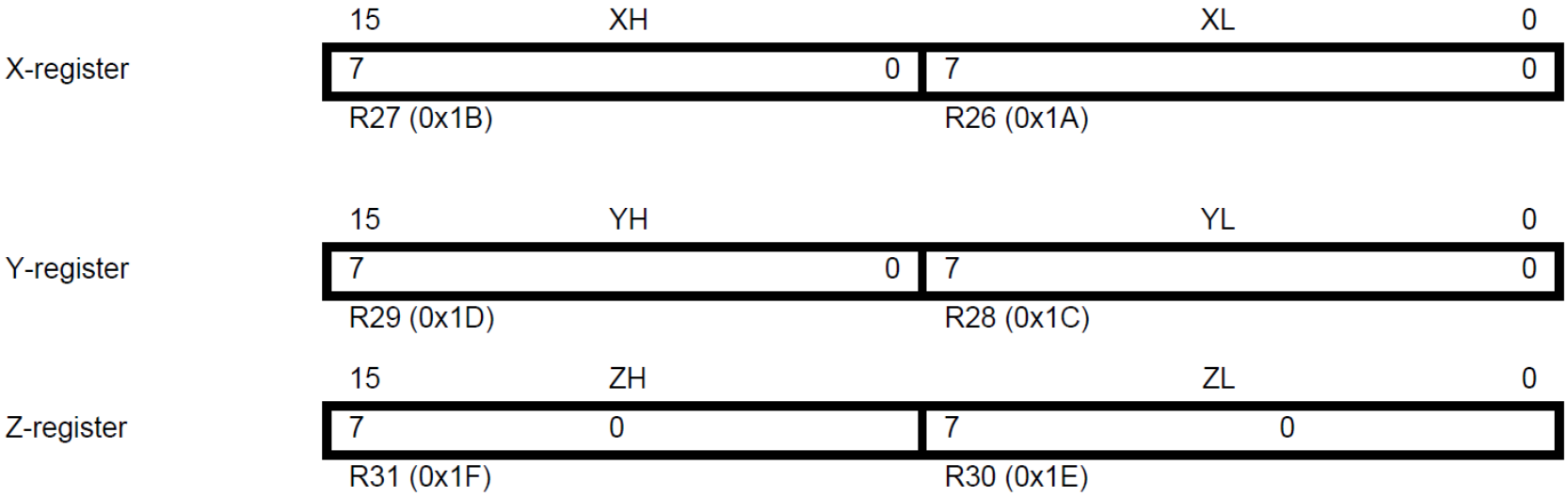
Registers

7	0	Addr.	
R0		0x00	
R1		0x01	
R2		0x02	
…			
R13		0x0D	
R14		0x0E	
R15		0x0F	
R16		0x10	
R17		0x11	
…			
R26		0x1A	X-register Low Byte
R27		0x1B	X-register High Byte
R28		0x1C	Y-register Low Byte
R29		0x1D	Y-register High Byte
R30		0x1E	Z-register Low Byte
R31		0x1F	Z-register High Byte

Registers R26-R31 play a special role in addressing.



Registers



The six 8-bit registers R26-R31 should be treated as three 16-bit registers, named X, Y, and Z.

Used for indirect addressing, see instructions LD and LDD in lecture 8.

Status register

The Status Register is an 8-bit register, not one of the 32 general purpose registers we have already discussed.

Its 8-bits are used as flags, and in the documentation of the AVR assembly have standard names.

I	T	H	S	V	N	Z	C	bit name
0	1	0	0	0	1	0	1	example bit value

In one of the usages of the status register, arithmetic and logical instructions may update some of the status register bits, depending on the result of the operation. This information can then be used by the next instruction (see SUBI instruction in lecture 9).

Overview

The ATmega328P microcontroller:

- main technical characteristics
- registers
- memory map
- I/O

Memory

Two main memory spaces (Harvard architecture), the Data Memory (SRAM and EEPROM) and the Program Memory (Flash).

The same essentially architecture comes in various memory sizes (and perhaps some additional capabilities).

Device	Flash	EEPROM	RAM	Interrupt Vector Size
ATmega48A	4KBytes	256Bytes	512Bytes	1 instruction word/vector
ATmega48PA	4KBytes	256Bytes	512Bytes	1 instruction word/vector
ATmega88A	8KBytes	512Bytes	1KBytes	1 instruction word/vector
ATmega88PA	8KBytes	512Bytes	1KBytes	1 instruction word/vector
ATmega168A	16KBytes	512Bytes	1KBytes	2 instruction words/vector
ATmega168PA	16KBytes	512Bytes	1KBytes	2 instruction words/vector
ATmega328	32KBytes	1KBytes	2KBytes	2 instruction words/vector
ATmega328P	32KBytes	1KBytes	2KBytes	2 instruction words/vector

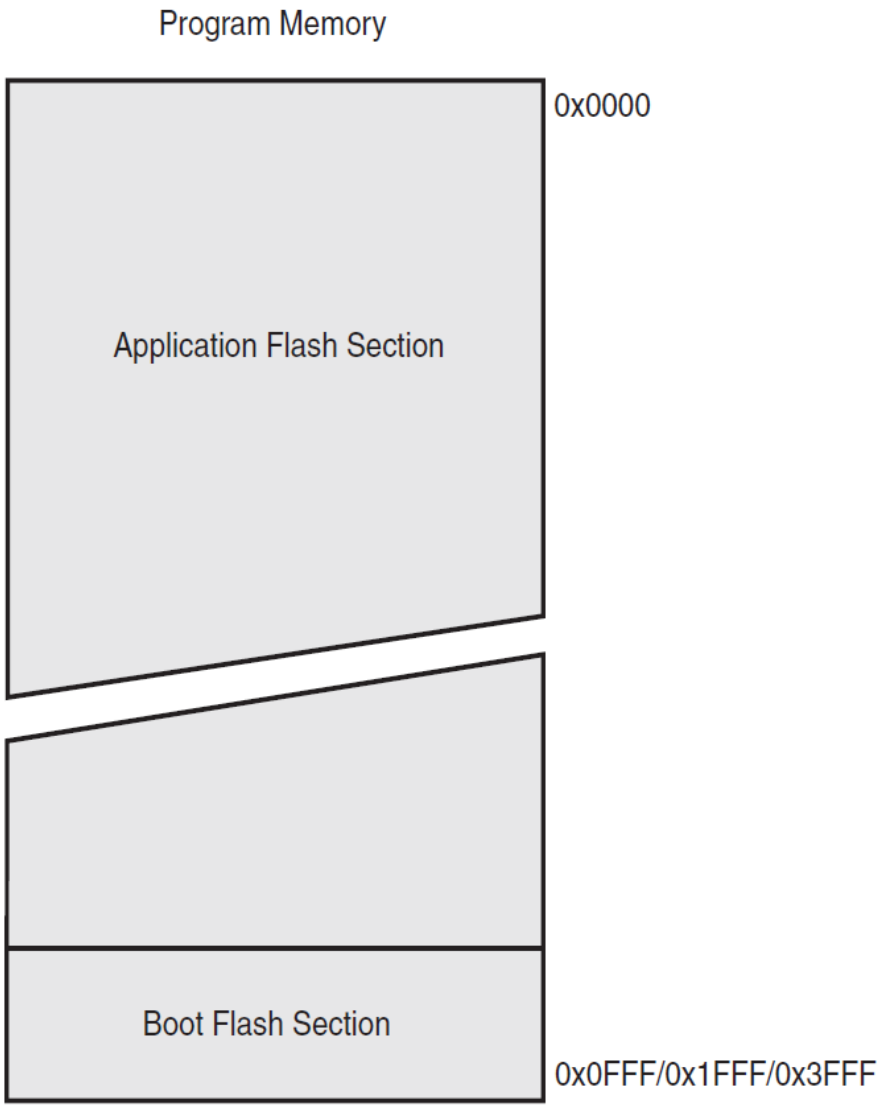
Program memory

Since AVR instructions are 16 or 32 bits wide, the Flash is organized as 2/4/8/16K x 16 (16K x 16 in 328P).

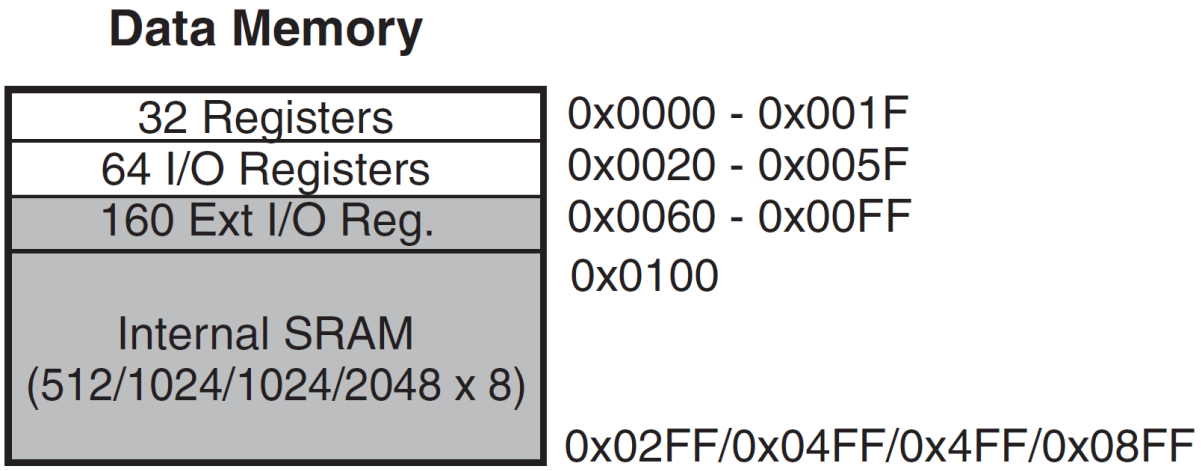
It is divided into two sections, Boot Loader Section and Application Program Section.

Arduino's Flash has an endurance of at least 10,000 write/erase cycles.

The Program Counter is 14 bits wide, thus addressing the 16K program memory locations.



Data memory (SRAM)



The Static Random Access Memory (SRAM) is divided into sections used as:

- mapping space of ALU registers
- I/O registers and extended I/O registers
- internal SRAM

EEPROM memory

1KB of EEPROM (**E**lectrically **E**rasable **P**rogrammable **R**ead-**O**nly **M**emory)

It is organized as a separate space for storing data.

Arduino's EEPROM has an endurance of at least 100,000 write/erase cycles.

In this lecture …

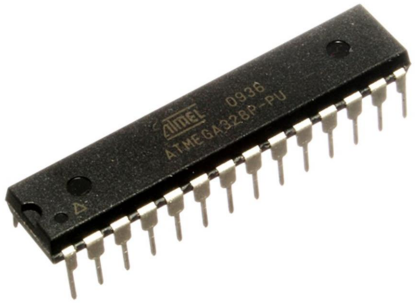
ATmega328P:

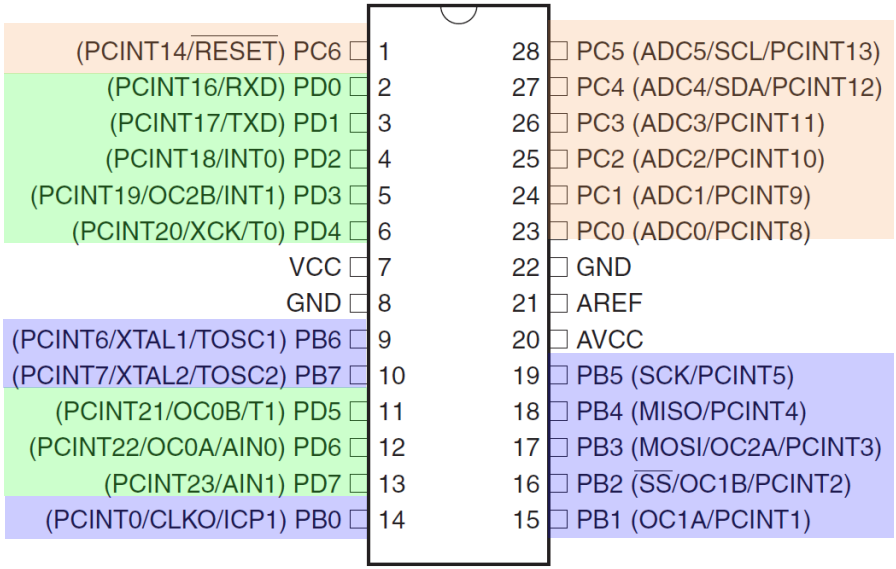
- main technical characteristics of
- registers
- memory map
- I/O

I/O ports

ATmega328P's I/O pins are grouped in three ports:

- B (digital)
- C (analog)
- D (digital)





Data Direction Registers

The direction of each pin of a port (input or output) is controlled by 8-bit Data Direction Registers: DDRB, DDRC, and DDRD.

Notice that we do not have to use the explicit memory locations of these registers. The assembler recognises the names DDRB, DDRC, and DDRD.

In fact, while in principle we can find their specific memory locations (they are just I/O registers specified by the manufacturer) and use them instead, that would affect the compatibility of our code with other AVR microcontrollers.

Example: setting pin direction

Whether a pin is used as input or output is controlled by a single bit:

We say a bit is set when is 1. Then, the corresponding pin is output.

A bit is cleared when is 0. Then, the corresponding pin is input.

We can control a pin's direction by manipulating single values of a register with the instructions SBI (set bit) and CBI (clear bit), see lecture 9.

```
SBI    DDRB, 4          ;set PB4     (declare pin PB4 as output)

CBI    DDRD, 2          ;clear PD2   (declare pin PD2 as input)
```

Reading input pin values

Assume that a pin has been declared as input pin. At any specific moment in the time, this pin has a certain status. Either there is an incoming electric signal, or not.

This information is recorded on the corresponding bit of the corresponding I/O register, PINB, PINC, and PIND.

Example: In the previous example we declared pin PD2 as input. The second bit (PIND,2) of the register PIND records the current status of that pin. Remember its value could change continuously.

Setting output pin values

Assume that a pin has been declared as output pin. At any specific moment in the time, we can set this pin to a certain status. Either we send an outcoming electric signal through that pin, or not.

This is controlled by the corresponding bit of the I/O registers PORTB, PORTC, and PORTD.

Example: In the previous example we declared pin PB4 as input. The second bit (PORTB,4) of the register PORTB controls the current status of that pin. By setting that bit, we send electric current through that pin. By clearing it, the electric current stops.

Next …

… addressing modes

and the

instruction set