

Lecture 01

# 임베디드시스템 설계 소개

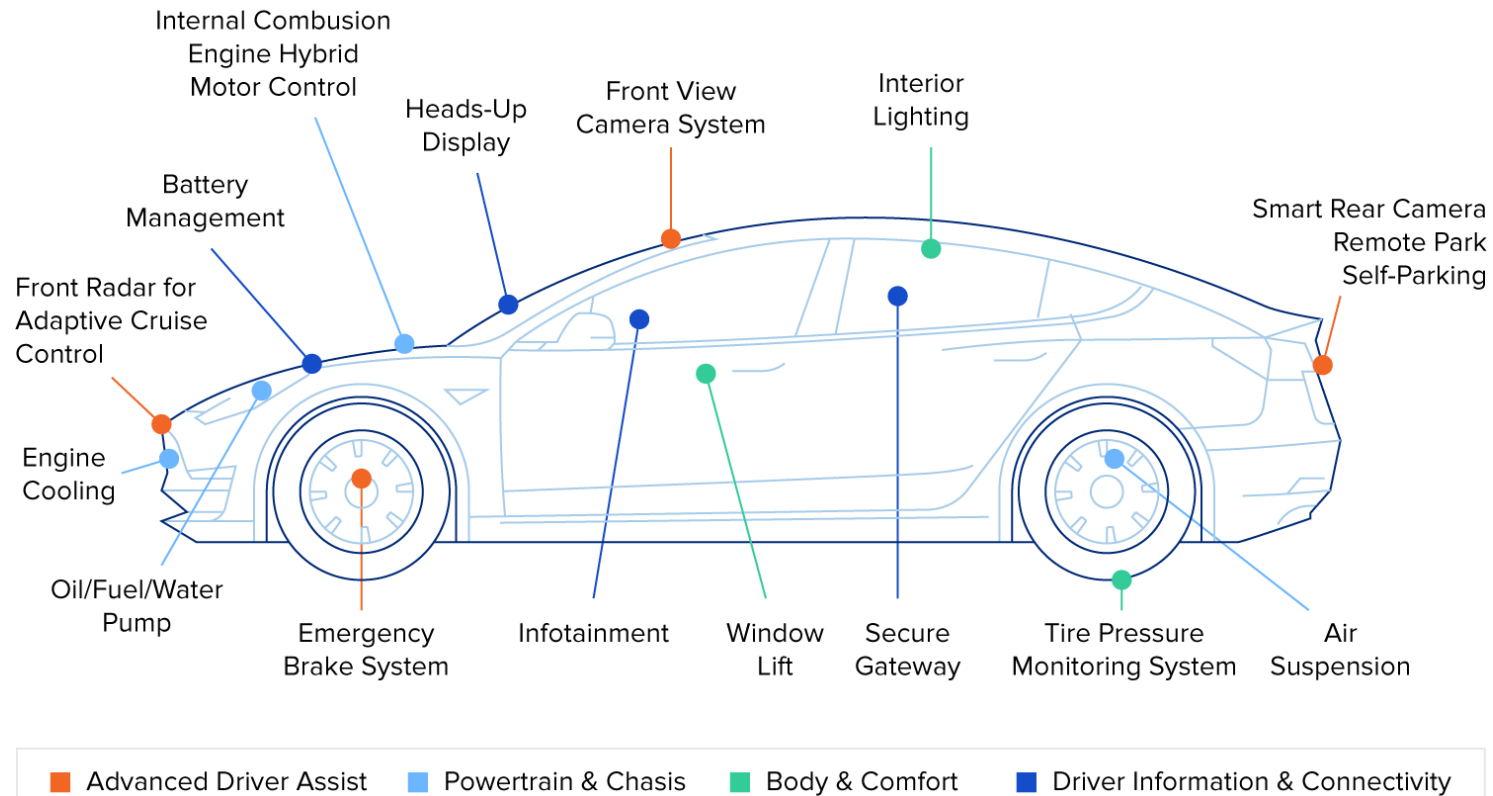
# 임베디드시스템

## ■ 일상생활 임베디드시스템



# 임베디드시스템

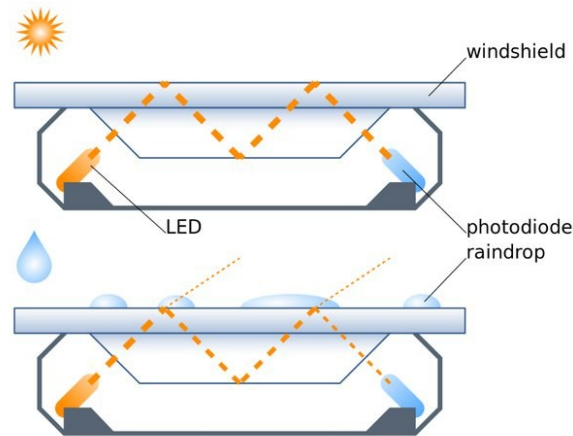
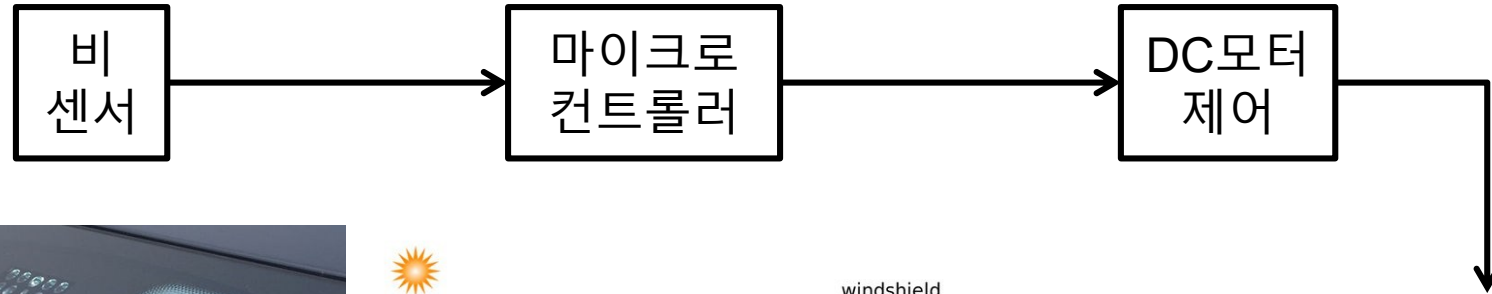
## ■ 자동차 임베디드시스템



\* Image source: Toptal

# 임베디드시스템

- 자동차 임베디드시스템 - 와이퍼 자동 제어





# 취업 관련

## 삼성전자 DS부문

### 영상 처리 알고리즘 설계

- AI Deep learning & Computer Vision 알고리즘 개발
- 이미지 센서, 멀티미디어 IP 용 ISP(Image Signal Processor) 알고리즘 개발
- 차세대 Sensor(DVS, SLAM 등) 알고리즘 개발
- 주요 연구 분야: Image Stabilization / WDR / Gamma Correction / Sensor compensation / Face verification / Noise Reduction / Demosaicing / Auto Focus / Auto Exposure / Auto White Balance

### [System LSI사업부] Camera S/W 개발

#### 직무정보

#### 참고사항

#### 수행업무

- Camera RTA (Real-time Algorithm) 솔루션 개발
  - 3A algorithms을 포함한 화질 솔루션 개발
  - Computervision에 기반한 화질 솔루션 개발 (Motion estimation, flicker detection, object tracking, etc.)
- Image quality engineer
  - 알고리즘 (RTA, ISP HW) deep understanding based 화질 튜닝 및 trouble shooting
- Camera System SW 개발
  - Android Camera HAL 개발
  - Linux Device Driver 개발
  - Firmware 개발

#### 지원자격

- 전공 관련
  - 전기전자, 컴퓨터공학 전공



# 취업 관련

## 현대자동차

### 임베디드 소프트웨어 개발 및 검증

채용시까지 2022-12-05 00:00 ~ 채용시까지



#### 📄 조직소개

- 우리 조직은 차량의 편의 시스템을 구성하는 요소인 모터, 스위치, 센서, 밸브 장치를 제어하는 표준화된 제어를 개발하고 있습니다.
- 사양 개발 및 검증 업무를 수행하며 하드 코드 개발로 제어기에 적용하는 업무를 하고 있습니다.

#### [함께 하고 싶은 분]

- 제어기술 전문성을 동료들과 잘 공유하고, 배려하는 성향을 가진 분
- 자기주도적이고 적극적인 태도와 효율적인 업무 수행을 추구하는 분

#### 📄 직무상세

- S/W 자체개발을 통한 S/W 단위/통합 및 HILs 검증 및 라이브러리 배포
- MATLAB을 활용한 사양 정합성 검증 및 배포

#### [SW 개발 및 검증]

- AUTOSAR 플랫폼 기반 Application S/W 개발
- S/W 정/동적 검증, 검증환경 구축 및 Test case 개발, S/W통합 및 HILs 검증
- 고장모드 도출, 안전설계 및 검증 업무 진행

#### [제어 사양 개발]

- 시스템 요구사항 및 아키텍처 개발
- 제어 사양서 표준화를 통한 업무 효율화 및 차종 사양서 개발

#### 👤 지원자격

- 학사 학위 이상 소지자
- 제어 공학 계열 (전기전자/로보틱스/컴퓨터/제어/IT 등) 전공자
- Application S/W 개발, 제어 사양 개발 및 검증 경험 보유자
- 임베디드 소프트웨어 경험 보유자
- C/C++ 활용 역량 보유자
- 최종합격 후, 회사가 지정하는 입사일에 입사 가능하신 분
- 해외여행에 결격 사유가 없는 분 (남성의 경우, 회사가 지정한 입사일까지 병역을 마쳤거나 면제되신 분)

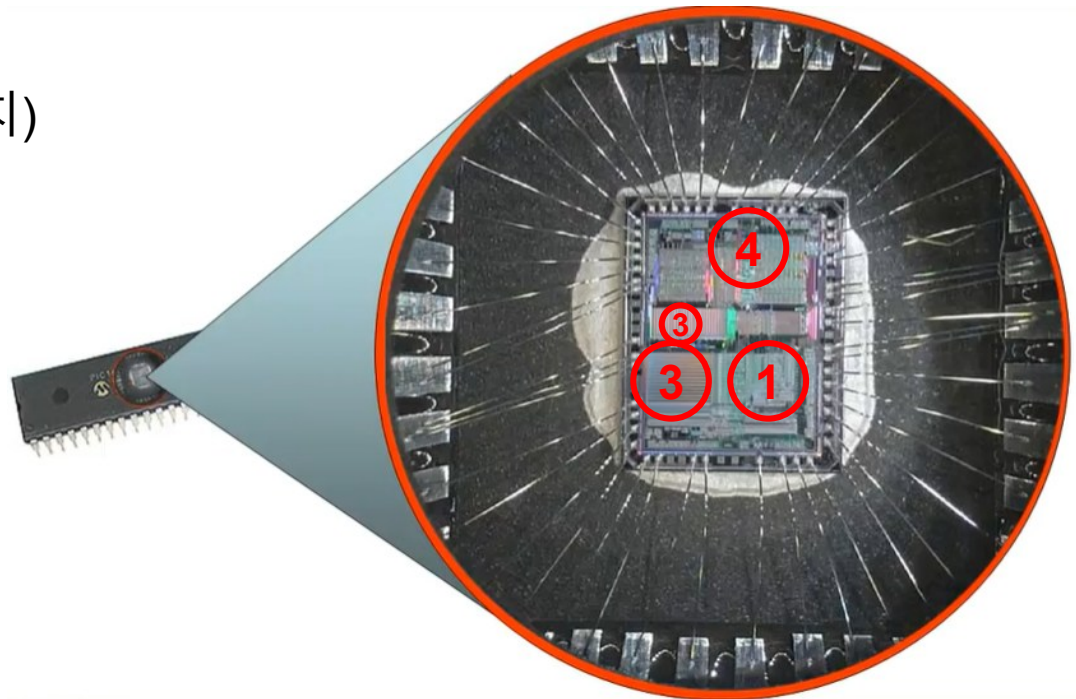
#### 📄 우대사항

- 임베디드 소프트웨어 관련 전공자
- 비즈니스 영어 활용 가능자
- 차량 공조 및 열관리 제어 요구사항서 개발 및 S/W 검증
- MATLAB, AUTOSAR
- SW아키텍처 설계 경험 보유자

# AVR 마이크로컨트롤러

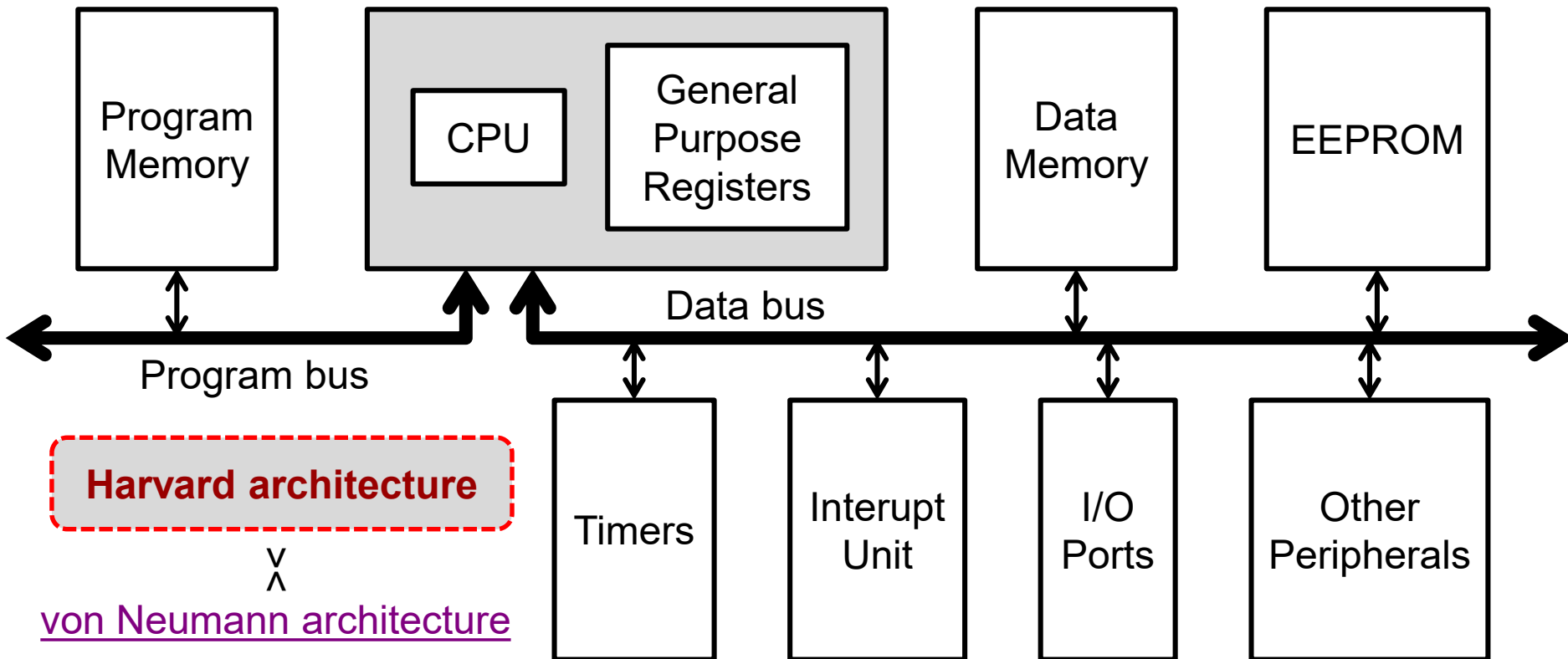
## ■ 마이크로컨트롤러 기본 요소

- ① Central Processing Unit (CPU)
- ② System clock
- ③ Memory
- ④ Peripherals (주변 장치)



# AVR 마이크로컨트롤러

- 일반적인 AVR 마이크로컨트롤러 블록도





# AVR 마이크로컨트롤러

- AVR 마이크로컨트롤러
  - (modified) Harvard architecture
  - 8-bit RISC(Reduced Instruction Set Computer)
- AVR families

	Flash size	Freq. (MHz)	Package	SRAM	EEPROM
<b>tinyAVR</b>	0.5~32 KB	1.6~20	6~32 pins	32~3072 bytes	64~512 bytes
<b>megaAVR</b>	4~256 KB	1.6~20	28~100 pins	256~16384 bytes	256~4096 bytes
<b>AVR Dx</b>	16~128 KB	20~24	14~64 pins	4~16 KB	512 bytes
<b>XMEGA</b>	16~256 KB	32	44~100 pins	1~32 KB	512~2048 bytes
<b>AVR32</b>	NA				

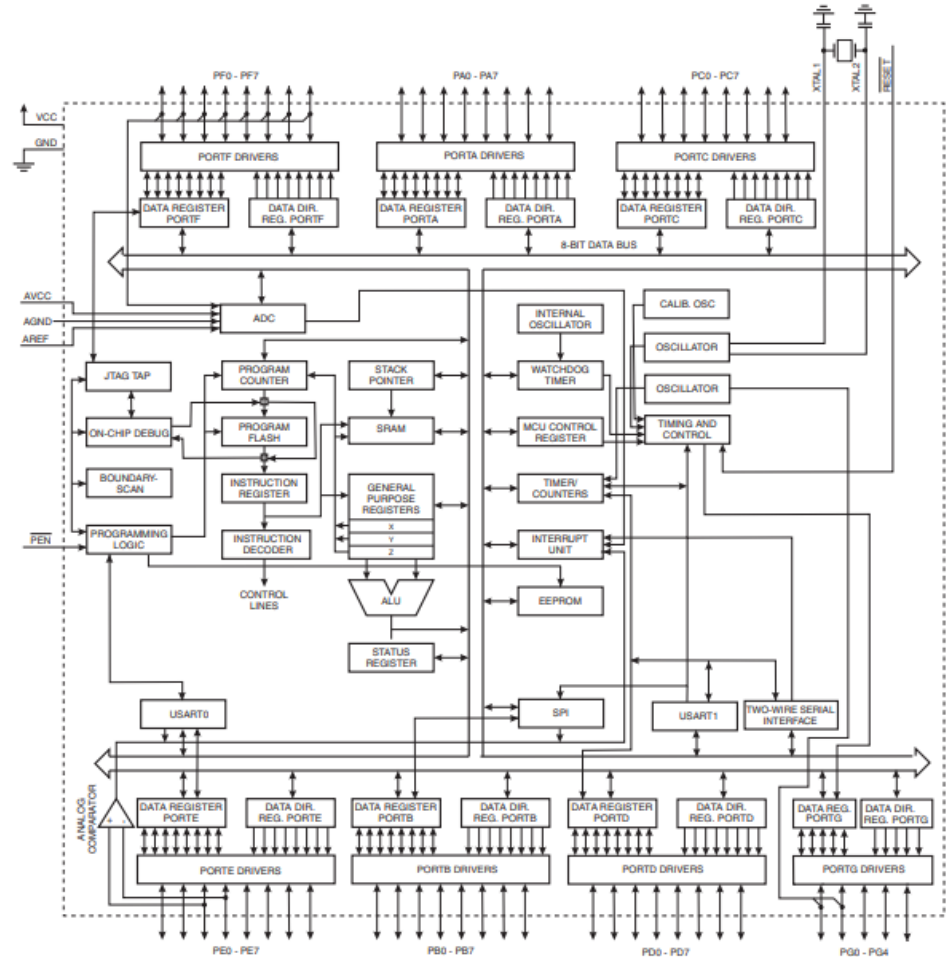
# AVR 마이크로컨트롤러



## ■ ATmega128

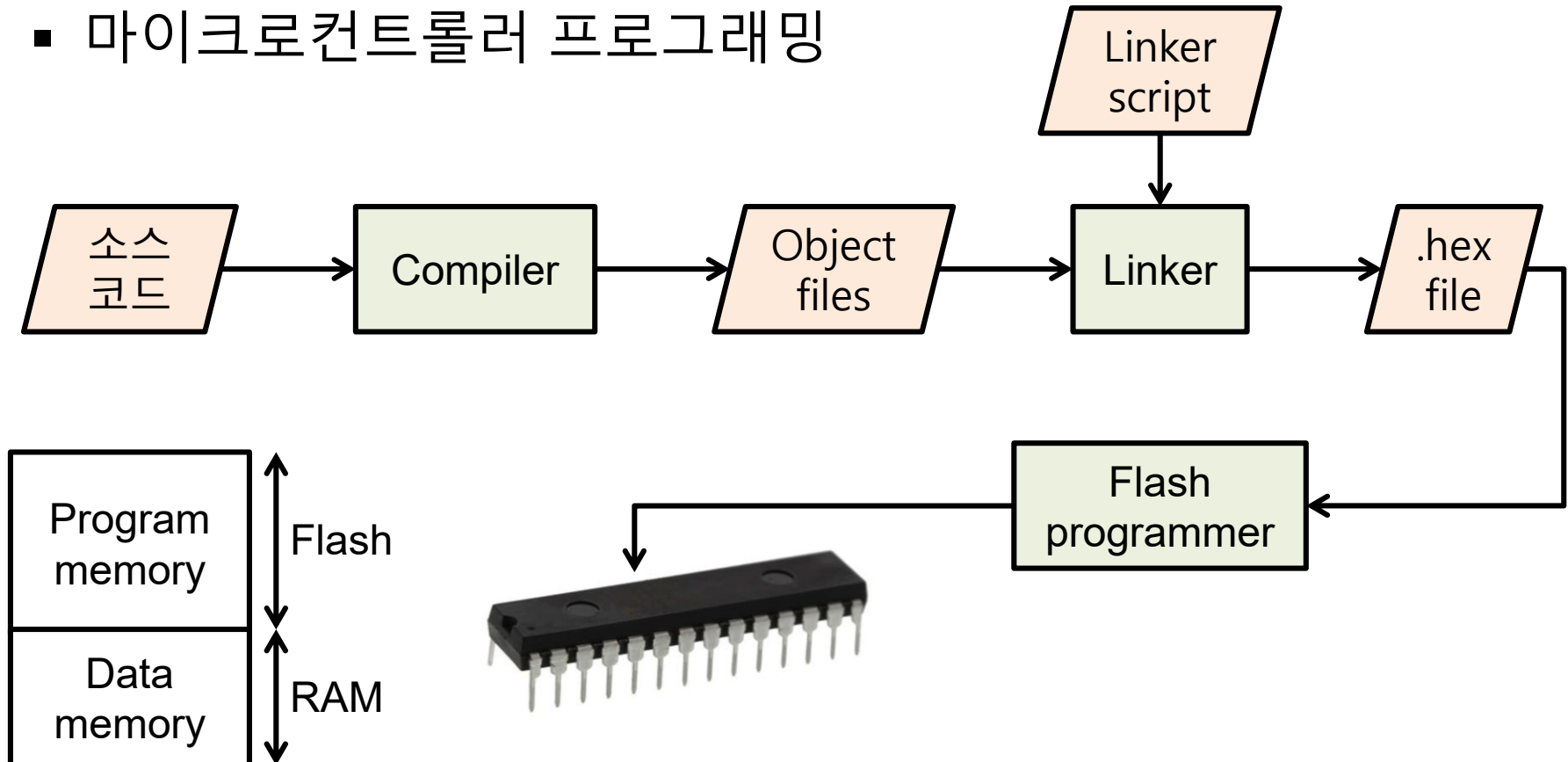
### Features

- High-performance, Low-power Atmel® AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 133 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers + Peripheral Control Registers
  - Fully Static Operation
  - Up to 16MIPS Throughput at 16MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 128Kbytes of In-System Self-programmable Flash program memory
  - 4Kbytes EEPROM
  - 4Kbytes Internal SRAM
  - Write/Erase cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C<sup>(1)</sup>
  - Optional Boot Code Section with Independent Lock Bits
  - In-System Programming by On-chip Boot Program
  - True Read-While-Write Operation
  - Up to 64Kbytes Optional External Memory Space
  - Programming Lock for Software Security
  - SPI Interface for In-System Programming
- QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix acquisition
  - Up to 64 sense channels
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - Two Expanded 16-bit Timer/Counters with Separate Prescaler, Compare Mode and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Two 8-bit PWM Channels
  - 6 PWM Channels with Programmable Resolution from 2 to 16 Bits
  - Output Compare Modulator
  - 8-channel, 10-bit ADC
    - 8 Single-ended Channels
    - 7 Differential Channels
    - 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Dual Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with On-chip Oscillator
  - On-chip Analog Comparator



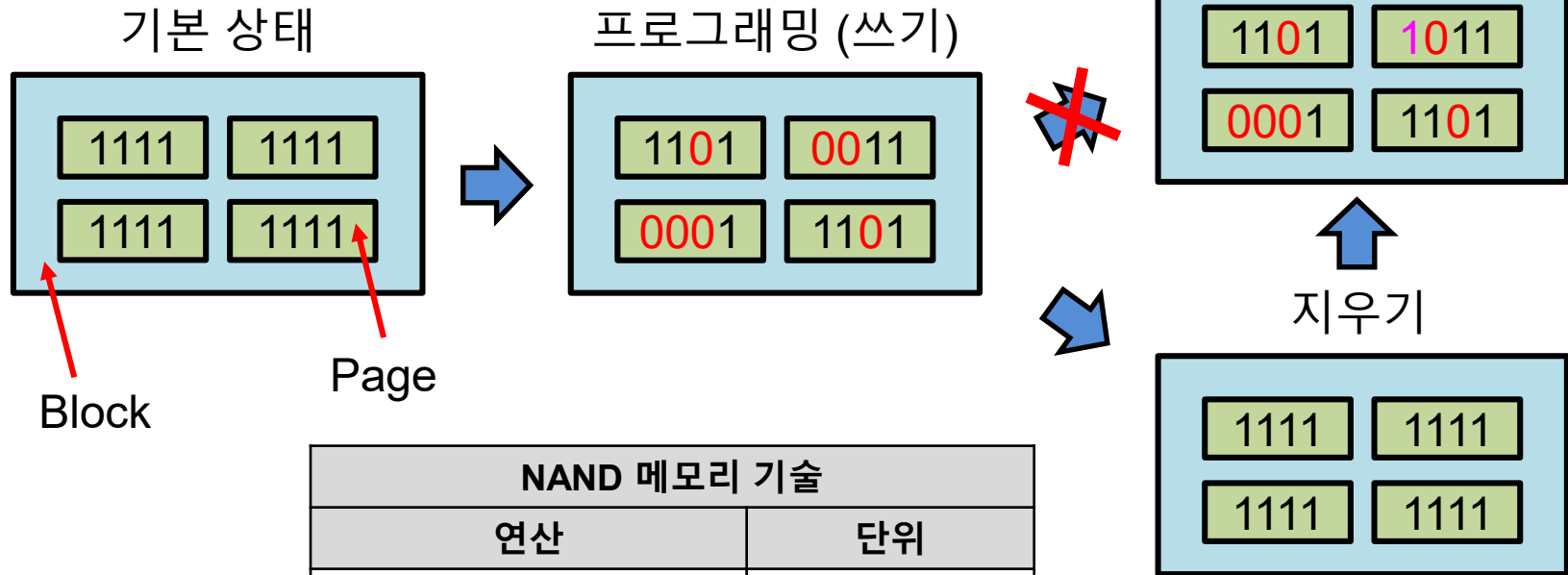
# AVR 마이크로컨트롤러

- 마이크로컨트롤러 프로그래밍



# AVR 마이크로컨트롤러

## ■ 마이크로컨트롤러 메모리

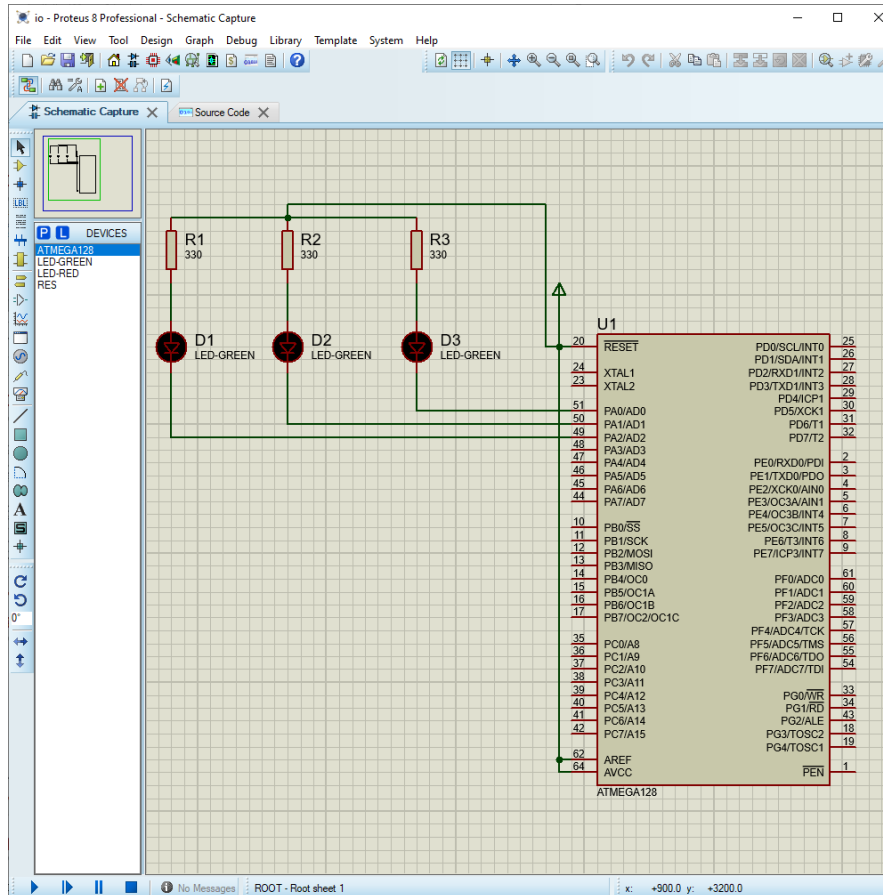


NAND 메모리 기술	
연산	단위
읽기	Page
프로그래밍 (쓰기)	Page
지우기	Block

# Proteus & Microchip Studio



## Proteus



## Microchip Studio

```
main.c  ASF Wizard  Advanced Mode  Quick Launch (Ctrl+Q)
File Edit View VAssist ASF Project Build Debug Tools Window Help
Release
main.c
/*
 * main.c
 * Created: 2/24/2024 10:42:24 PM
 * Author: DatNgo
 */

#include <xc.h>
#include <avr/io.h>
#include <util/delay.h>

void delay(int d);

int main(void)
{
    DDRA = 0xff;
    while(1)
    {
        //TODO: Please write your application code
        PORTA = 0x07;
        delay(2000);
        PORTA = 0x00;
        delay(2000);
    }
}

void delay(int d) {
```

# Proteus & Microchip Studio



## ▪ Blinking LEDs

```
#include <xc.h>           // compiler
#include <avr/io.h>       // 입출력 포트 (DDRA 및 PORTA 사용)
#include <util/delay.h>   // _delay_ms() 함수 사용

int main(void) {
    DDRA = 0xff;         // PORTA 출력 모드 설정
    while (1) {
        PORTA = 0x07;    // PA0, PA1, PA2를 1로 설정 (LED on)
        _delay_ms(2000); // delay 2s
        PORTA = 0x00;    // PA0, PA1, PA2를 0로 설정 (LED off)
        _delay_ms(2000); // delay 2s
    }
}
```

# Proteus & Microchip Studio

## Memory-mapped I/O

```
avr/im0128.h { /* Data Direction Register, Port A */
                #define DDRA _SFR_IO8(0x1A)
                /* Data Register, Port A */
                #define PORTA _SFR_IO8(0x1B)
```

```
avr/sfr_defs.h { #define _SFR_IO8(io_addr) _MMIO_BYTE((io_addr)+_SFR_OFFSET)
                  #define _MMIO_BYTE(mem_addr) (*(volatile uint8_t*)(mem_addr))
                  #ifndef __SFR_OFFSET
                  # if __AVR_ARCH__ >= 100
                  # define __SFR_OFFSET 0x00
                  # else
                  # define __SFR_OFFSET 0x20
                  # endif
                  #endif
                  #endif
```

DDRA = 0x1A **혹은** DDRA = 0x3A  
 PORTA = 0x1B **혹은** PORTA = 0x3B

# Proteus & Microchip Studio

## Memory-mapped I/O

ATmega128

DDRA = 0x1A 혹은 DDRA = 0x3A  
 PORTA = 0x1B 혹은 PORTA = 0x3B

Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
\$61	DDRF	DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0	88
\$60	Reserved	--	--	--	--	--	--	--	--	
\$3F (\$5F)	SREG	I	T	H	S	V	N	Z	C	10
\$3E (\$5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	13
\$3D (\$5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	13
\$3C (\$5C)	XDIV	XDIVEN	XDIV6	XDIV5	XDIV4	XDIV3	XDIV2	XDIV1	XDIV0	36
\$3B (\$5B)	RAMPZ	--	--	--	--	--	--	--	RAMPZ0	13
\$3A (\$5A)	EICRB	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	90
\$39 (\$59)	EIMSK	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	91
\$38 (\$58)	EIFR	INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	INTF0	91
\$37 (\$57)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	108, 138, 158
\$36 (\$56)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	108, 140, 159
\$35 (\$55)	MCUCSR	SRE	SRW10	SE	SM1	SM0	SM2	IVSEL	IVCE	30, 44, 63
\$34 (\$54)	MCUCSR	JTD	--	--	JTRF	WDRF	BORF	EXTRF	PORF	53, 254
\$33 (\$53)	TCCR0	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	103
\$32 (\$52)	TCNT0	Timer/Counter0 (8 Bit)								105

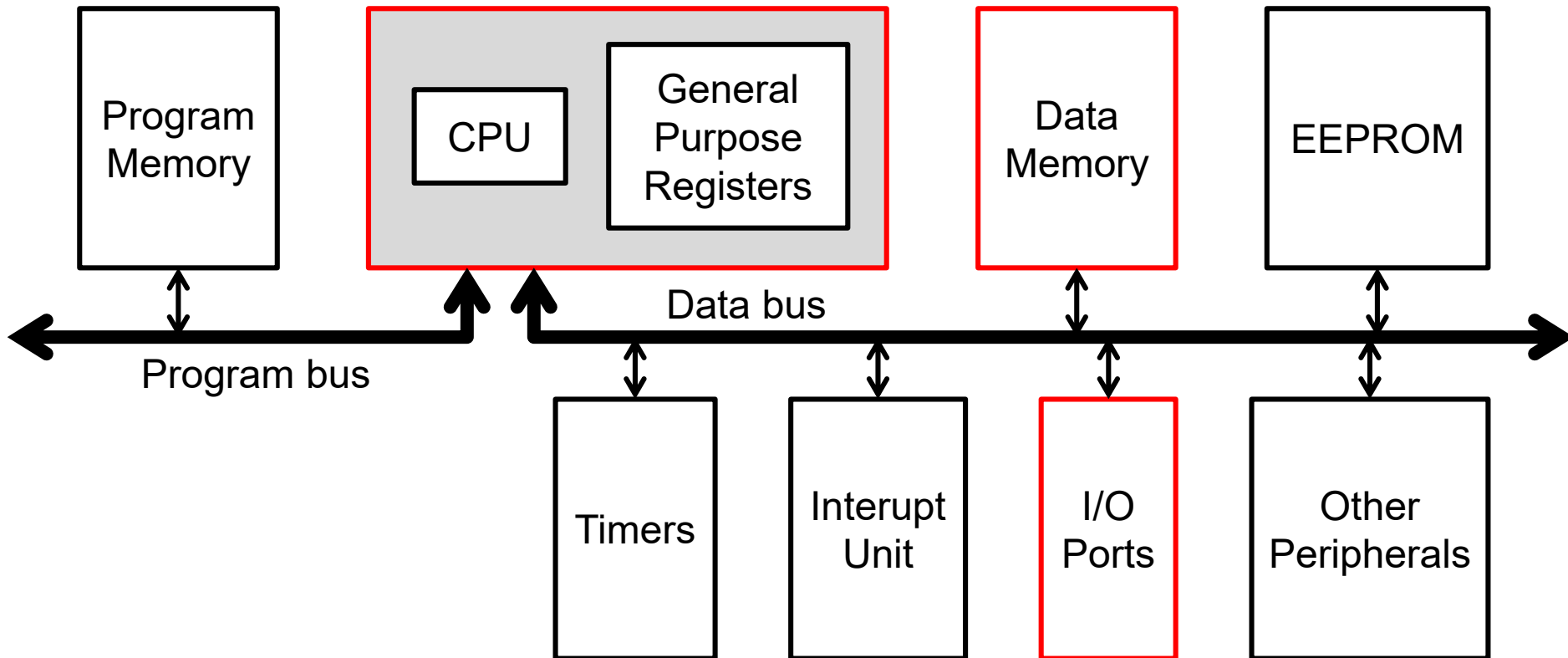
\$1B (\$3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	86
\$1A (\$3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	86

\$20 (\$40)	SFIO	TSM	--	--	--	--	ACME	PUD	PSR0	PSR31	72, 108, 144, 227
\$1F (\$3F)	EEARH	--	--	--	--	--	--	EEPROM Address Register High			20
\$1E (\$3E)	EEARL	EEPROM Address Register Low Byte									20
\$1D (\$3D)	EEDR	EEPROM Data Register									21
\$1C (\$3C)	EEDR	--	--	--	--	--	EERIE	EEMWE	Eewe	EERE	21
\$1A (\$3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	86	
\$19 (\$39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	86	



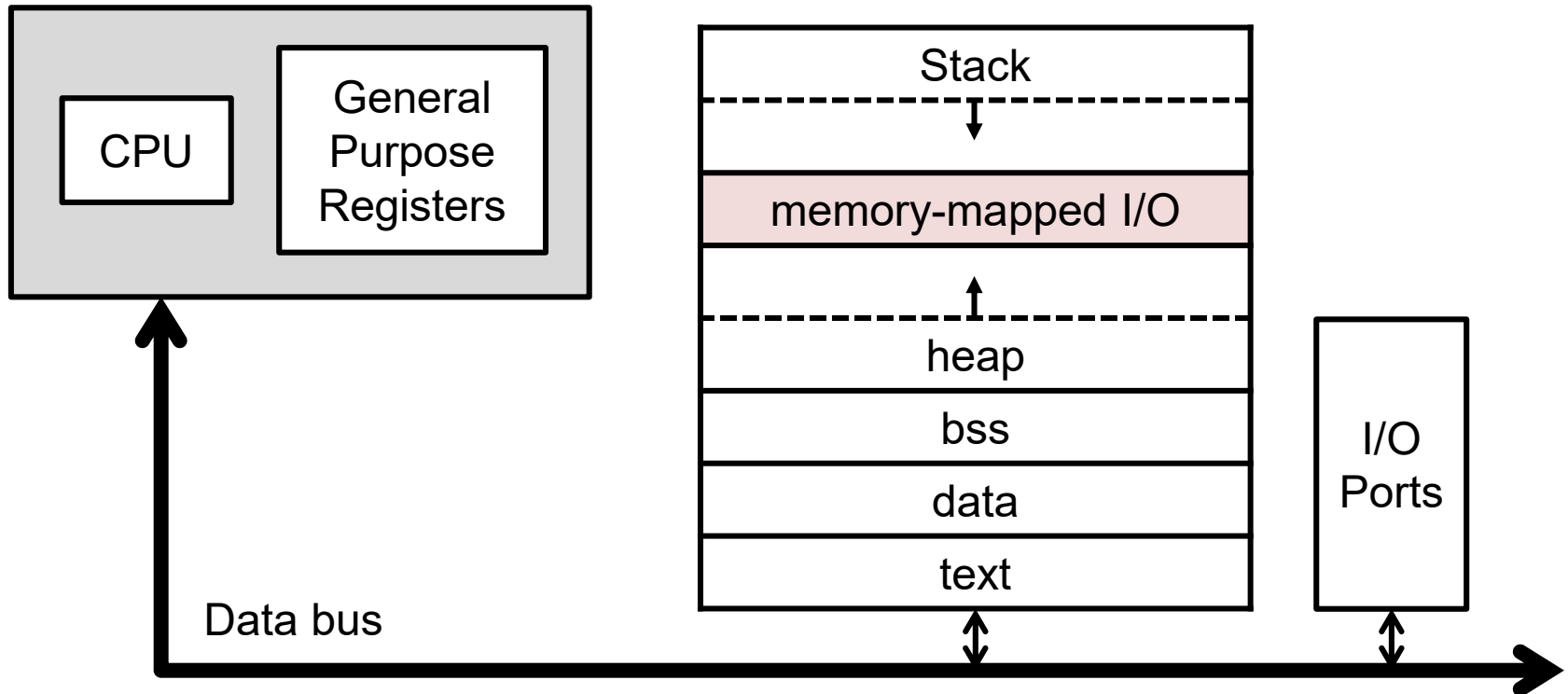
# Proteus & Microchip Studio

- Memory-mapped I/O



# Proteus & Microchip Studio

- Memory-mapped I/O



# Appendix

- von Neumann architecture

