



C basic language

임베디드스쿨 2기

Lv1과정

2021. 05. 14

김효창

명령어

AVRDUDE는 AVR 마이크로 컨트롤러의 ROM 및 EEPROM 콘텐츠를 다운로드/업로드/조작 등을 도와주는 유틸리티

Atmel Studio & CodevisionAVR은 Linux 환경을 지원하지 않으므로,
Linux 환경에서 Hex 파일을 AVR microcontrollers에 다운로드 하기 위해 사용

```
sudo avrdude -c avrisp2 -p m328p -U flash:w:blink_test.hex
```

sudo : 임시 관리자 권한 접근

avrdude : AVR 의 ROM 및 EEPROM 접근

-c : 프로그래머

avrisp2 : 사용 중인 프로그래머 이름

-p : 부품 part 은 atmega328p (m328p)

m328p : 사용 중인 board

-U : <filename> 을 읽기/쓰기/확인

프로그램 해야 하는 메모리는 플래쉬(flash) : 쓰기(w): 파일명은 ○ ○.hex

blink_test.hex : hex 파일 이름

[avrdude 매뉴얼 참고..](#)

PORTB / DDRB / PINB

13.4.2 PORTB – The Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

13.4.3 DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

13.4.4 PINB – The Port B Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

SREG (status Register)

6.3.1 SREG – AVR Status Register

The AVR status register – **SREG** – is defined as:

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – I: Global Interrupt Enable**

The global interrupt enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the global interrupt enable register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

용어 정리

PORTB

0 → output 0 [V] (Low)

1 → output 5 [V] (High)

PORT 는 출력 값으로 사용 (PIN 은 입력 값으로 사용)

DDRB

0 → Input

1 → Output

각 Bit 의 입력 또는 출력 모드를 설정

PINB

PIN 으로부터 데이터를 입력 받는 레지스터

DDRB 가 입력으로 설정되어 있으면 High / Low 상태를 읽는다.

용어 정리

SREG (status Register)

인터럽트를 활성화하려면 global interrupt Enable 를 (1) 로 설정해야 한다.

Global Interrupt Enable → sei () ; , SREG |= (1 << 7)

Global Interrupt Disable → cli () ; , SREG |= (0 << 7)

전역 인터럽트 활성화 레지스터를 지운 경우 개별 인터럽트 활성화 설정과 관계없이 모든 인터럽트가 활성화되지 않는다.

외부 인터럽트는 입출력 포트의 기능을 사용하는 것이 아니므로 DDR 레지스터는 설정하지 않는다.

Polling

부모님이 오늘 집에 오기로 했는데 도착하기 전까지 수시로 집 밖으로 나가서 확인하는 것을 반복.

Interrupt

부모님이 오늘 집에 오기로 했는데 집 안에서 게임하다가 부모님이 도착해서 초인종을 눌렀을 때 밖으로 나가서 확인.

용어 정리

인터럽트는 스택 함수 호출과 비슷하다.

- 일반적인 작업(task) 실행
- 인터럽트 발생(요청)
- 수행 작업 정지 (복귀할 주소 값 저장)
- 인터럽트 벡터 테이블 → ISR 주소 값으로 점프
- 인터럽트 서비스 루틴 실행
- ISR 종료 후 복귀
- 멈췄던 작업 실행

S/W 인터럽트 : 마이크로프로세서 내부에서 발생 , 타이머 / 카운터 등

H/W 인터럽트 : 마이크로프로세서 외부에서 발생 , 스위치 / 센서 등

인터럽트 사용 시 코드에 #include <avr/interrupt.h> 선언

인터럽트 서비스 루틴은 가능하면 변수 값 처리만 하도록 권장.

```
ISR ( OO_vector )  
{  
    인터럽트 서비스 루틴 코드 삽입  
}
```

AND / OR / XOR / NOT

AND

스위치 또는 센서의 상태를 입력 처리할 때 사용
Bit 값이 0 으로 출력할 때 사용

OR

시프트 연산 또는 출력 처리할 때 사용

XOR

반전 출력할 때 사용
Bit 가 같다 = 0 , 다르다 = 1

Features

- High performance, low power AVR[®] 8-bit microcontroller
- Advanced RISC architecture
 - 131 powerful instructions – most single clock cycle execution
 - 32 × 8 general purpose working registers
 - Fully static operation
 - Up to 16MIPS throughput at 16MHz
 - On-chip 2-cycle multiplier
- Speed grade:
 - 0 to 8MHz at 2.7 to 5.5V (automotive temperature range: –40°C to +125°C)
 - 0 to 16MHz at 4.5 to 5.5V (automotive temperature range: –40°C to +125°C)

8.8 External Clock

To drive the device from an external clock source, XTAL1 should be driven as shown in [Figure 8-4](#). To run the device on an external clock, the CKSEL fuses must be programmed to “0000” (see [Table 8-14](#)).

Table 8-14. Crystal Oscillator Clock Frequency

Frequency	CKSEL3..0
0 to 16MHz	0000

Delay 사용 방법

28.3 DC Characteristics

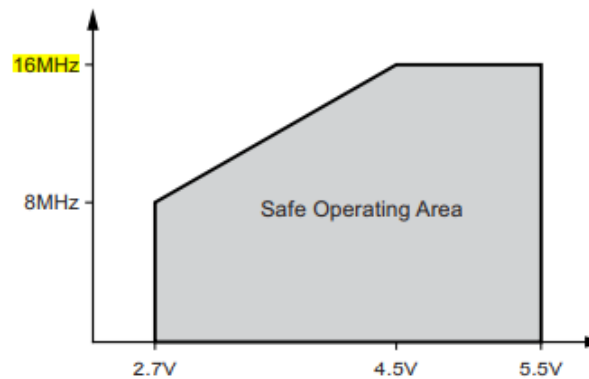
$T_A = -40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$, $V_{CC} = 2.7\text{V}$ to 5.5V (unless otherwise noted)

Parameter	Condition	Symbol	Min.	Typ. ⁽²⁾	Max.	Units
Power supply current ⁽¹⁾	Active 4MHz, $V_{CC} = 3\text{V}$	I_{CC}		1.5	2.4	mA
	Active 8MHz, $V_{CC} = 5\text{V}$			5.2	10	mA
	Active 16MHz, $V_{CC} = 5\text{V}$			9.2	14	mA
	Idle 4MHz, $V_{CC} = 3\text{V}$			0.25	0.6	mA
	Idle 8MHz, $V_{CC} = 5\text{V}$			1.0	1.6	mA
	Idle 16MHz, $V_{CC} = 5\text{V}$			1.9	2.8	mA
Power-down mode ⁽³⁾	WDT enabled, $V_{CC} = 3\text{V}$	I_{CC}			44	μA
	WDT enabled, $V_{CC} = 5\text{V}$				66	μA
	WDT disabled, $V_{CC} = 3\text{V}$				40	μA
	WDT disabled, $V_{CC} = 5\text{V}$				60	μA

- Notes: 1. Values with [Section 9.10 "Minimizing Power Consumption" on page 36](#) enabled (0xFF).
2. Typical values at 25°C .
3. The current consumption values include input leakage current.

28.4 Speed Grades

Figure 28-1. Maximum Frequency



Delay 사용 방법

#include <util/delay.h>

F_CPU 정의되지 않으면 경고 발생.

외부 크리스탈과 동일한 값을 F_CPU 에 define 해야 한다.

define F_CPU 16000000L

: 16 MHz 사용 , L 은 자료형 long 4 Byte 의미 16 / 00 / 00 / 00

_delay_ms(2500);

: 2.5초 지연 , us 도 사용 가능

기타

while (1) ; while (1) 다음 세미콜론 필수 , 중괄호 작성 시는 세미콜론 생략.
(조건식) : 0 = 실행 중지, 1 = 계속 수행.

volatile : 인터럽트 서비스 루틴에서 사용되는 변수를 선언할 때 사용
volatile unsigned char switch;

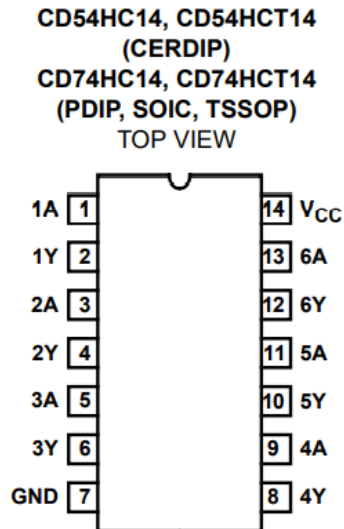
unsigned 는 양수 , 0 만을 가진다. 0 ~ 255 사이의 값
음수를 표현하기 위해서는 unsigned 를 사용하지 않는다.

74HC14

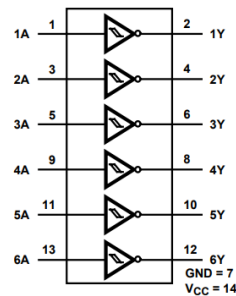
74HC14 는 74HC04 + Schmitt Trigger 를 추가한 6 개의 게이트가 들어 있는 회로

Schmitt Trigger

- 입력 값의 노이즈 때문에 출력 값이 수시로 변화한다.
- 상위 값 4V , 하위 값 3V 가정하면 4V 가 초과 되어야 High 상태가 되며 3V 미만일 때 Low 상태가 된다. (3.9 , 3.7 , 3.5 , 3.3V 등으로 감소해도 High 상태를 유지한다.)
- 상위 값과 하위 값의 차를 Hysteresis 전압이라고 부른다.
- 사인파를 구형파 상태로 만든다.



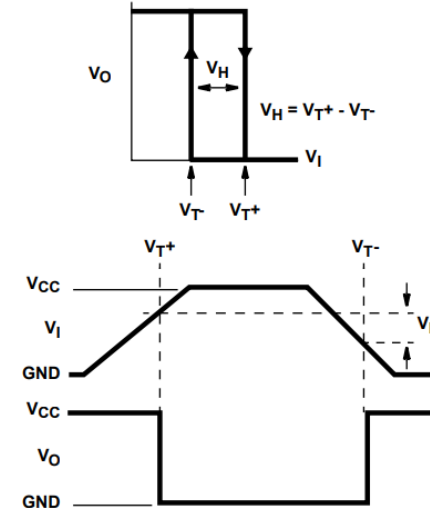
Functional Diagram



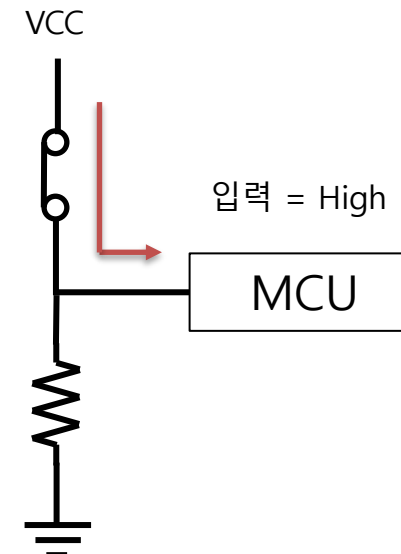
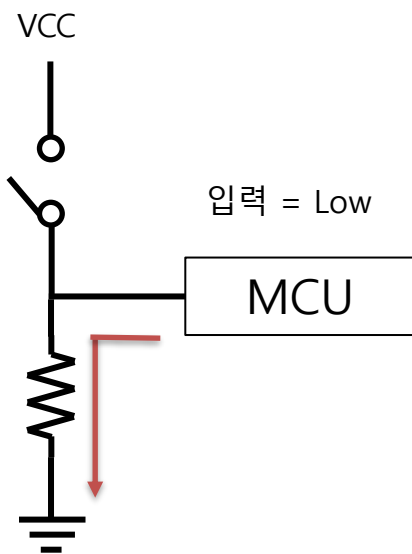
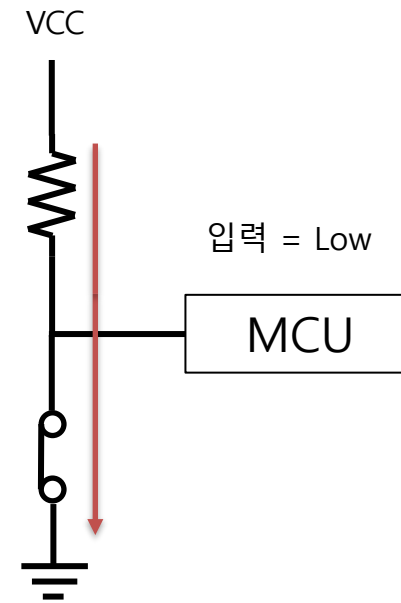
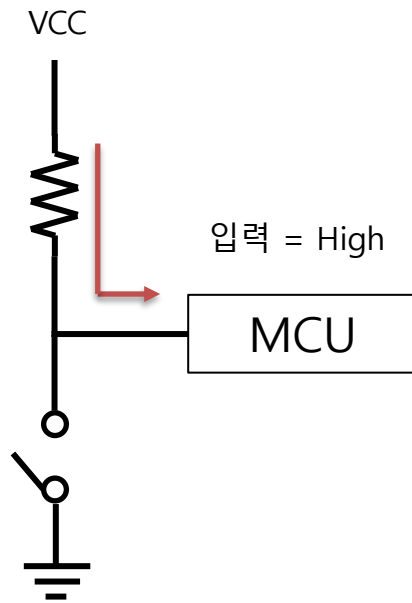
TRUTH TABLE

INPUT (A)	OUTPUT (Y)
L	H
H	L

H= High Level
L= Low Level



MCU 상태



Pin Change Interrupt Control Register

12.2.4 PCICR – Pin Change Interrupt Control Register

Bit	7	6	5	4	3	2	1	0	
(0x68)	–	–	–	–	–	PCIE2	PCIE1	PCIE0	PCICR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7..3 - Res: Reserved Bits**

These bits are unused bits in the Atmel® ATmega328P, and will always read as zero.

- **Bit 2 - PCIE2: Pin Change Interrupt Enable 2**

When the PCIE2 bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt 2 is enabled. Any change on any enabled PCINT23..16 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI2 interrupt vector. PCINT23..16 pins are enabled individually by the PCMSK2 register.

- **Bit 1 - PCIE1: Pin Change Interrupt Enable 1**

When the PCIE1 bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt 1 is enabled. Any change on any enabled PCINT14..8 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI1 interrupt vector. PCINT14..8 pins are enabled individually by the PCMSK1 register.

- **Bit 0 - PCIE0: Pin Change Interrupt Enable 0**

When the PCIE0 bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt 0 is enabled. Any change on any enabled PCINT7..0 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI0 interrupt vector. PCINT7..0 pins are enabled individually by the PCMSK0 register.

Pin Change Mask Register

12.2.6 PCMSK2 – Pin Change Mask Register 2

Bit	7	6	5	4	3	2	1	0	
(0x6D)	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	PCMSK2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7..0 – PCINT23..16: Pin Change Enable Mask 23..16**

Each PCINT23..16-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT23..16 is set and the PCIE2 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT23..16 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

12.2.7 PCMSK1 – Pin Change Mask Register 1

Bit	7	6	5	4	3	2	1	0	
(0x6C)	–	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	PCMSK1
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – Res: Reserved Bit**

This bit is an unused bit in the Atmel® ATmega328P, and will always read as zero.

- **Bit 6..0 – PCINT14..8: Pin Change Enable Mask 14..8**

Each PCINT14..8-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT14..8 is set and the PCIE1 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT14..8 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

12.2.8 PCMSK0 – Pin Change Mask Register 0

Bit	7	6	5	4	3	2	1	0	
(0x6B)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7..0 – PCINT7..0: Pin Change Enable Mask 7..0**

Each PCINT7..0 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is set and the PCIE0 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

Interrupt Vector

11.1 Interrupt Vectors in ATmega328P

Table 11-1. Reset and Interrupt Vectors in ATmega328P

Vector No.	Program Address	Source	Interrupt Definition
1	0x0000	RESET	External pin, power-on reset, brown-out reset and watchdog system reset
2	0x002	INT0	External interrupt request 0
3	0x0004	INT1	External interrupt request 1
4	0x0006	PCINT0	Pin change interrupt request 0
5	0x0008	PCINT1	Pin change interrupt request 1
6	0x000A	PCINT2	Pin change interrupt request 2
7	0x000C	WDT	Watchdog time-out interrupt

PCICR / PCMSK

PCICR

핀의 상태가 변화하면 인터럽트 발생. ($L \rightarrow H$, $H \rightarrow L$)

PCIE0 비트가 설정되고 (1) , 상태 레지스터 (SREG)의 I 비트가 설정 (1)되면 핀 변경 인터럽트 0 이 활성화된다..

활성화 된 PCINT7 ~ 0 핀의 변경은 인터럽트를 발생시킵니다.

핀 변경 인터럽트 요청의 해당 인터럽트는 PCI0 인터럽트 벡터에서 실행됩니다.

(데이터시트 상에 해당 범주를 지원한다면 가능하고 지원하지 않는다면 불가능합니다.)

PCINT 0 ~ 7 중에 신호가 변화하면 PCINT0_vector 호출

PCINT 8 ~ 14 중에 신호가 변화하면 PCINT1_vector 호출

PCINT 16 ~ 23 중에 신호가 변화하면 PCINT2_vector 호출

PCINT7 ~ 0 핀은 PCMSK0 레지스터에 의해 개별적으로 활성화된다.

PCMSK

PCINT7 ~ 0 Bit 는 해당 I / O 핀에서 핀 변경 인터럽트를 활성화할지 여부를 선택한다.

PCINT7 ~ 0 이 설정되고 PCICR의 PCIE0 비트가 설정되면 해당 I / O 핀에서 핀 변경 인터럽트가 활성화.

PCINT7 ~ 0 이 해제되면 해당 I / O 핀의 핀 변경 인터럽트가 비활성화된다.

Interrupt 설정 방법

STEP 1 : Turn on Pin Change Interrupts

PCINT를 설정하려면 PCICR 을 구성해야 한다.

모든 인코더를 PCINT23 : 16으로 라우팅 할 것이므로 PCIE2를 활성화 한다.

이를 위해 코드 라인: PCICR |= 0x04 을 포함.

여러 가지 방법으로 할 수 있지만 다른 비트는 변경하지 않고 핀은 1로 설정하는 것이 좋다.

STEP 2: Pin Selection

PCI2에 매핑 된 모든 핀은 해당 ISR을 실행하지만 핀 변경 마스크 레지스터에서 마스크 된 비트를 설정 한 경우에만 해당.

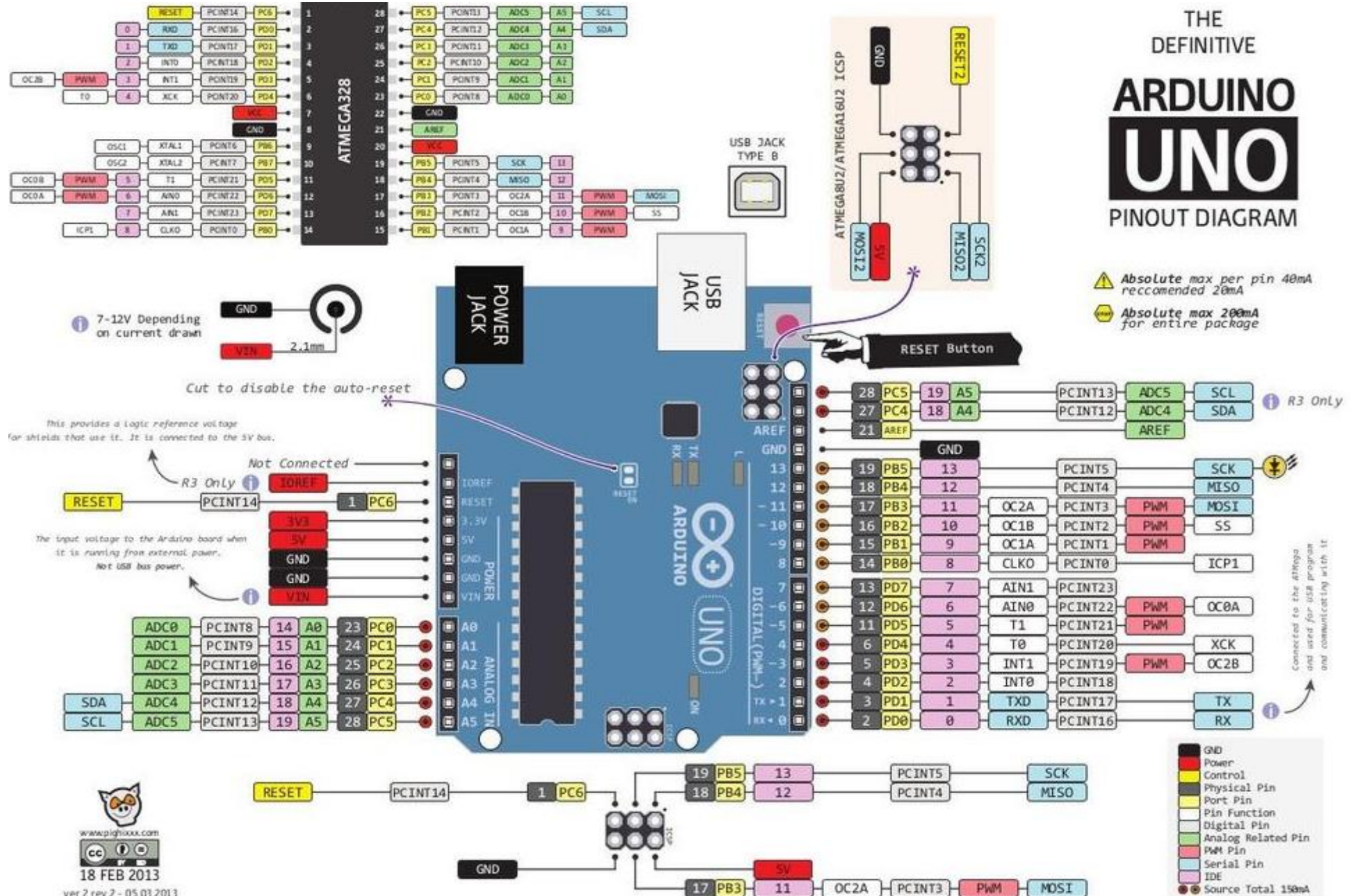
PCI2에 매핑 된 핀 8개 중 6개만 사용하면 나중에 필요할 경우에도 다른 아날로그-인 핀을 사용할 수 있다. PCINT 16:21에 대한 인코더만 배선하므로, 다음 코드 라인을 포함하면 됩니다: PCMSK2 |= 0x3F.

STEP 3: Write the ISR

ISR을 작성할 때마다 가능한 짧게 유지하고 변수를 선언할 때는 최적화되지 않도록 변수 유형을 휘발성 유형으로 만들어야 합니다.

ISR을 정의하려면 다음을 포함 : ISR(PCINT2_vect){} // PCINT16-PCINT23.

Pin Map



인터럽트 코드

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

```
int main (void)
{
    DDRB &= ~( 1 << DDB5 );
    PORTB |= ( 1 << PORTB4 );

    PCICR |= ( 1 << PCIE0 )
    PCMSK0 |= ( 1 << PCINT5 )

    sei ( );
    while (1);
    return 0;
}
```

```
ISR ( PCINT0_vect )
{
    PORTB ^= 0x10;
}
```

Port clear 해당 비트는 무조건 0, 입력으로 받겠다.
OR : 덮어쓰기 , 있다 = 변함없다 , 없다 = 반영

pin change interrupt 활성화
어떤 pin 을 사용할 것인지 설정

sei () 로 전체 인터럽트 활성화

ISR은 연결된 해당 Port의 Pin에서 조건이
충족 될 때마다 호출됩니다
해당 Pin 의 입력 상태(신호 레벨) 변화에 따라
ISR(PCINT0_vect) 로 즉시 이동해 함수가 실행된다.
13번 입력 핀(PCINT5)이 변화 할 때마다
12번 핀의 LED 출력을 반전한다.

인터럽트 코드

```
ISR ( PCINT0_vect )  
{  
    PORTB ^= 0x10;  
}
```

버튼을 누를 때마다 toggle

Bit	7	6	5	4	3	2	1	0
PORTB	0	0	0	0	0	0	0	0
0x10	0	0	0	1	0	0	0	0
XOR 결과 값 (1)	0	0	0	1	0	0	0	0
0x10	0	0	0	1	0	0	0	0
XOR 결과 값 (2)	0	0	0	0	0	0	0	0

회로 측정

그림 1. VCC 와 GND 사이의 전압

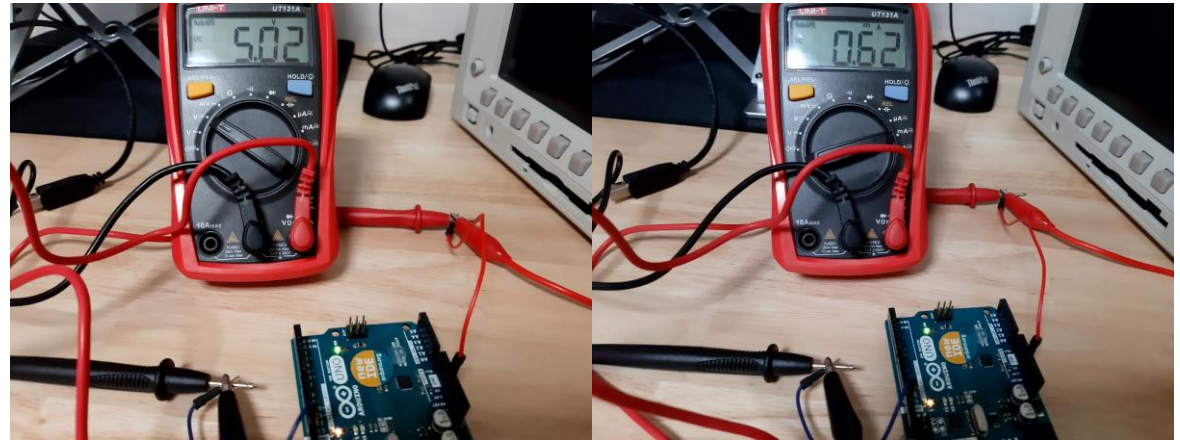
: 5.02 [V]

그림 2. VCC 와 GND 사이의 전류

: 0.6 [mA]

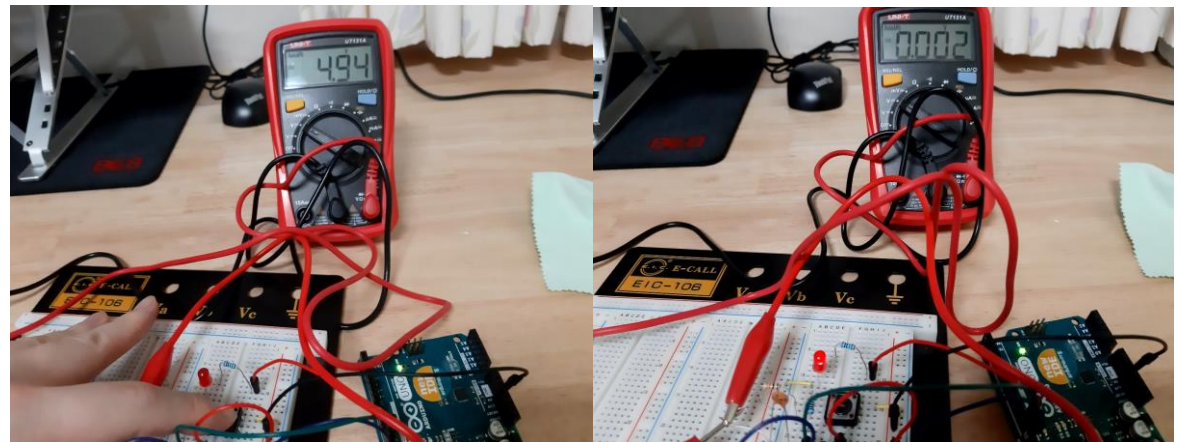
그림 3~4. 74HC14 의 Pin 2 와 GND 사이의 전압

- 스위치 ON → 4.94 [V]
- 스위치 OFF → 0.002 [V]



1

2



3

4

회로 측정

그림 1.

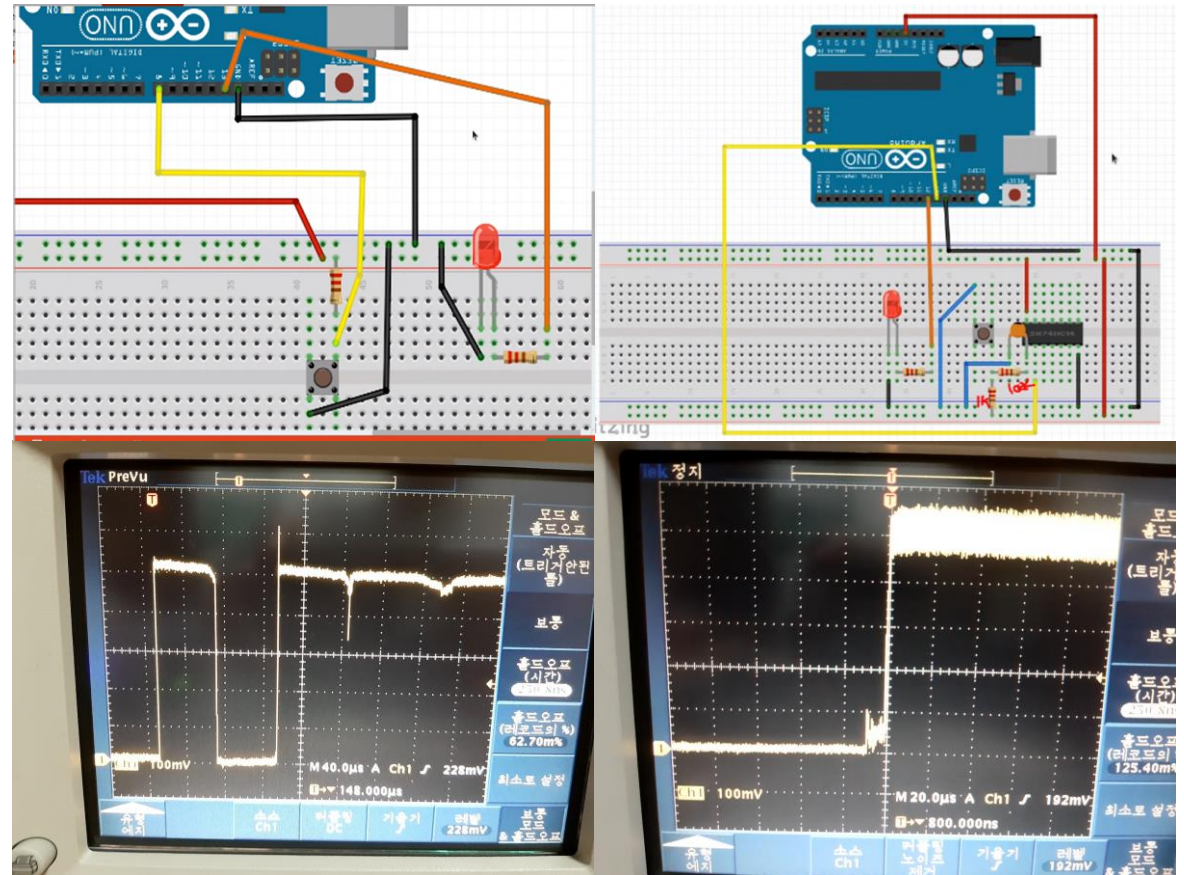
chattering : 스위치 접점이 닫히거나 열리는 순간에 기계적인 진동에 의해 매우 짧은 시간 안에 붙었다가 떨어지는 것을 반복하는 현상

그림 2.

RC 필터를 활용하여 파형이 정상적으로 표시된다.

C : 세라믹 콘덴서

(RC 필터 A4 용지 작성 중)



1

2

회로 측정

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

```
int main (void)
{
    DDRB &= ~( 1 << DDB5 );
    PORTB |= ( 1 << PORTB4 );

    PCICR |= ( 1 << PCIE0 )
    PCMSK0 |= ( 1 << PCINT5 )

    sei ( );
    while (1);
    return 0;
}
```

```
ISR ( PCINT0_vect )
{
    PORTB ^= 0x10;
}
```


파이프라인

파이프라인이 왜 필요한지부터 알아야하지 않을까요 ? 여기서 어떻게 확장되는지 궁극적으로 무엇을 하려고 하는지 등등요

기본적인 파이프라인에 대해 조사해보도록 합니다. 추가적으로 현재 **ARM** 아키텍처에 적용되는 **5단계 파이프라인**과 **9단계 파이프라인**에 대해 추가적인 조사를 해보는 것입니다. 그리고 과거 프로젝트 내용 보고 프로젝트 제안서 제출하는 것 까지를 이번주 숙제 범위로 잡겠습니다.

파이프라인 장점 구현한 것.

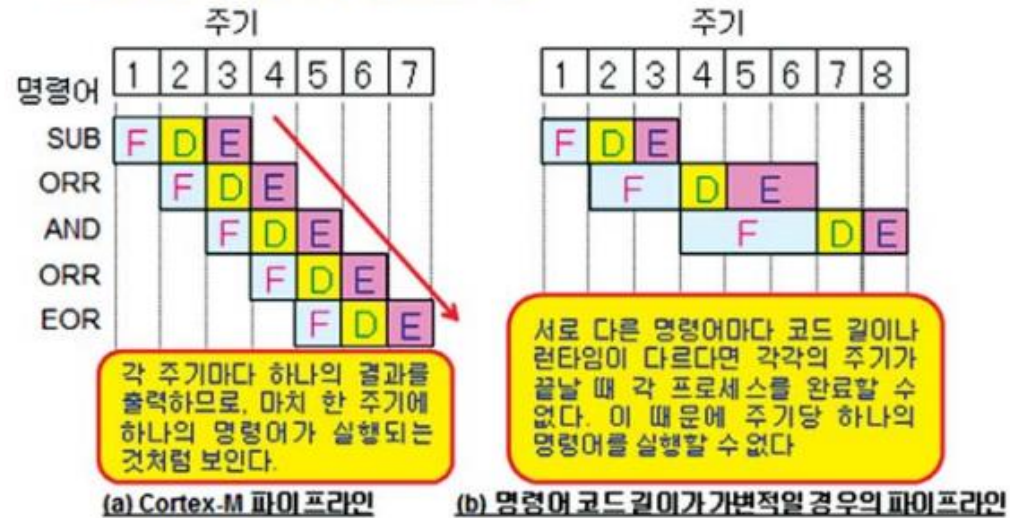
동작 속도가 빠르다. (해석 속도가 빠르고 여러 개의 명령어를 처리하기에 적합)
파이프라인과 슈퍼스칼라 기술을 쓰면 멀티태스킹이 가능.
하드웨어를 대폭 단순화 및 효율화 시킬 수 있다. (전력 소모가 적다)
가격경쟁력 측면에서 우위를 확보한다.

C source 는 Flash 메모리에 저장한다.
Flash 에 있는 데이터를 처리하기 위해 메모리에 접근해서 읽어야 한다. (Fetch 단계)
읽어온 데이터를 기계어로 번역 (Decode 단계)
기계어로 번역한 것을 실행 (Execute 단계)

파이프라인

명령어는 다음 3 단계로 실행된다.

- 1 단계: 명령어를 플레시메모리로부터 가져온다 (F).
- 2 단계: 명령어를 디코딩한다 (D).
- 3 단계: 명령어를 실행하고 결과를 저장한다 (E).



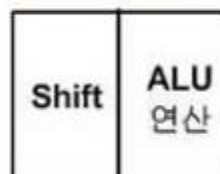
FETCH



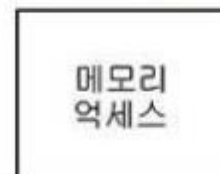
DECODE



EXECUTE



MEMORY



WRITE



파이프라인

IF는 메모리에서 명령어를 읽어오고 PC(program counter) 혹은 IP(instruction pointer)를 다음 명령어를 지시한다.

DE는 명령어를 해독. 비트단위로 쪼개서 어떤 instruction 인지 보고 그에 해당하는 제어신호를 보내 instruction을 처리. 또한 operand로 들어온 레지스터를 read한다.

EX는 산술(덧,뺀,곱,나눗셈), 논리(AND, OR, NOT) 등을 수행. coprocessor(보조프로세서)를 이용해 캐시, MMU 등을 컨트롤하거나 SIMD, 부동소수점 연산을 수행.

MEM은 메모리 접근을 수행. Memory에 R/W하는 동작은 여기서 실행.

WB은 레지스터 쓰기를 수행. Mem단계에서 읽어온 메모리의 값을 레지스터에 쓰거나, EX 단계에서 add 등을 이용해 연산된 결과를 레지스터에 저장할 때 사용.

파이프라인

1. 명령어 인출 상태

FETCH1 : $AR \leftarrow PC$

FETCH2 : $DR \leftarrow M$, $PC \leftarrow PC + 1$

FETCH3 : $IR \leftarrow DR[7..6]$, $AR \leftarrow DR[5..0]$

AC : 8비트 누산기

AR : 16비트 주소 레지스터

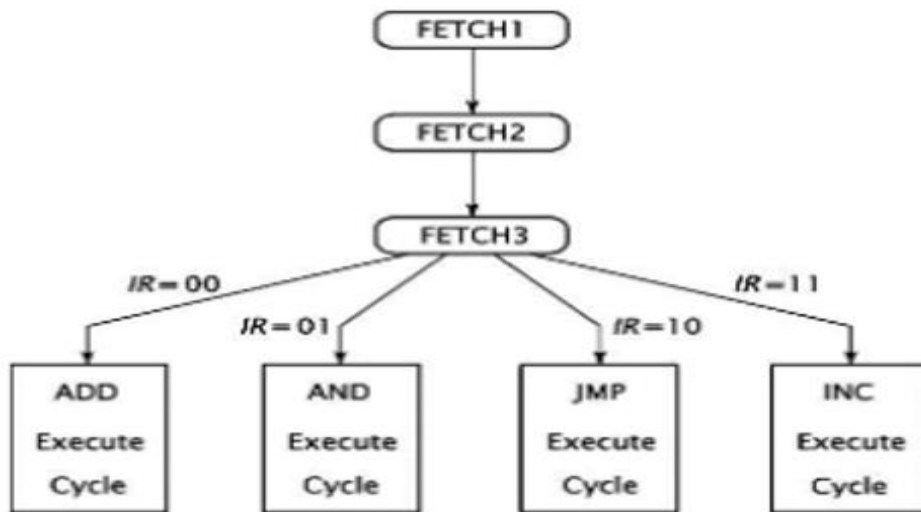
PC : 16비트 프로그램 카운터

DR : 8비트 데이터 레지스터

IR : 8비트 명령어 레지스터

TR : 8비트 임시 레지스터

2. 인출된 명령어를 해독하고 실행 루틴(어떤 작업을 정의한 명령어의 그룹) 결정



파이프라인

PC : Program Counter

다음에 인출할 명령어의 주소를 가지고 있는 레지스터

각 명령어가 인출된 후에는 자동으로 일정 크기(한 명령어 길이)만큼 증가
분기 (branch) 명령어가 실행되는 경우에는 목적지 주소로 갱신

AC : Accumulator

데이터를 일시적으로 저장하는 레지스터

레지스터의 크기는 CPU 가 한 번에 처리할 수 있는 데이터 비트 수 (단어 길이)

IR : Instruction Register

가장 최근에 인출된 명령어 코드가 저장되어 있는 레지스터

파이프라인 성능 저하 요인

파이프라인 하드웨어를 단순화하기 위해 모든 명령어가 stage 들을 모두 통과해야 한다.

파이프라인의 클럭은 처리 시간이 가장 오래 걸리는 stage 를 기준으로 한다.

분기 명령어가 실행되면 미리 인출하여 처리하던 명령어들이 무효된다.

파이프라인 논문

프로그램 카운터 RPC를 사용하면 명령어 캐시가 CPU에서 주소를 전송하지 않고도 연속 명령어에 액세스할 수 있다.

분기 또는 예외가 발생한 경우에만 명령 메모리에 CPU의 주소가 필요하다.

프로세서 구현의 상대적 성능 수치는 일반적으로 MIPS (초당 수백만 명령)로 표현되며, 주기 시간 Tcycle과 명령당 평균 사이클 수에 (CPI) 반비례한다. [average number of Cycles per Instruction]

스칼라 RISC 프로세서에 대한 CPI는 기본적으로 하나의 명령 사이클과 파이프 라인 위험으로 인한 평균 가능한 사이클 수입니다.

예를 들어, 분기 및 로드 페널티, 캐시 누락 후 지연 주기 등이 있다.

명령어 당 필요한 사이클의 수(CPI) 를 줄일 수 있다.

프로그램 흐름을 정확하게 예측함으로써 잘못 예측된 분기로 인해 발생할 수 있는 파이프라인 플러시를 최소화 한다.

기능 콜이 있을 때마다 복귀 주소가 하드웨어 스택으로 들어간다.

하드웨어가 기능 복귀 명령을 인식하면, 주소 값이 복귀 스택에서 튀어나와 예측된 복귀 주소로 사용. 정확한 복귀 주소가 확보되면 파이프라인 끝에서 해당 예측을 할 수 있다.

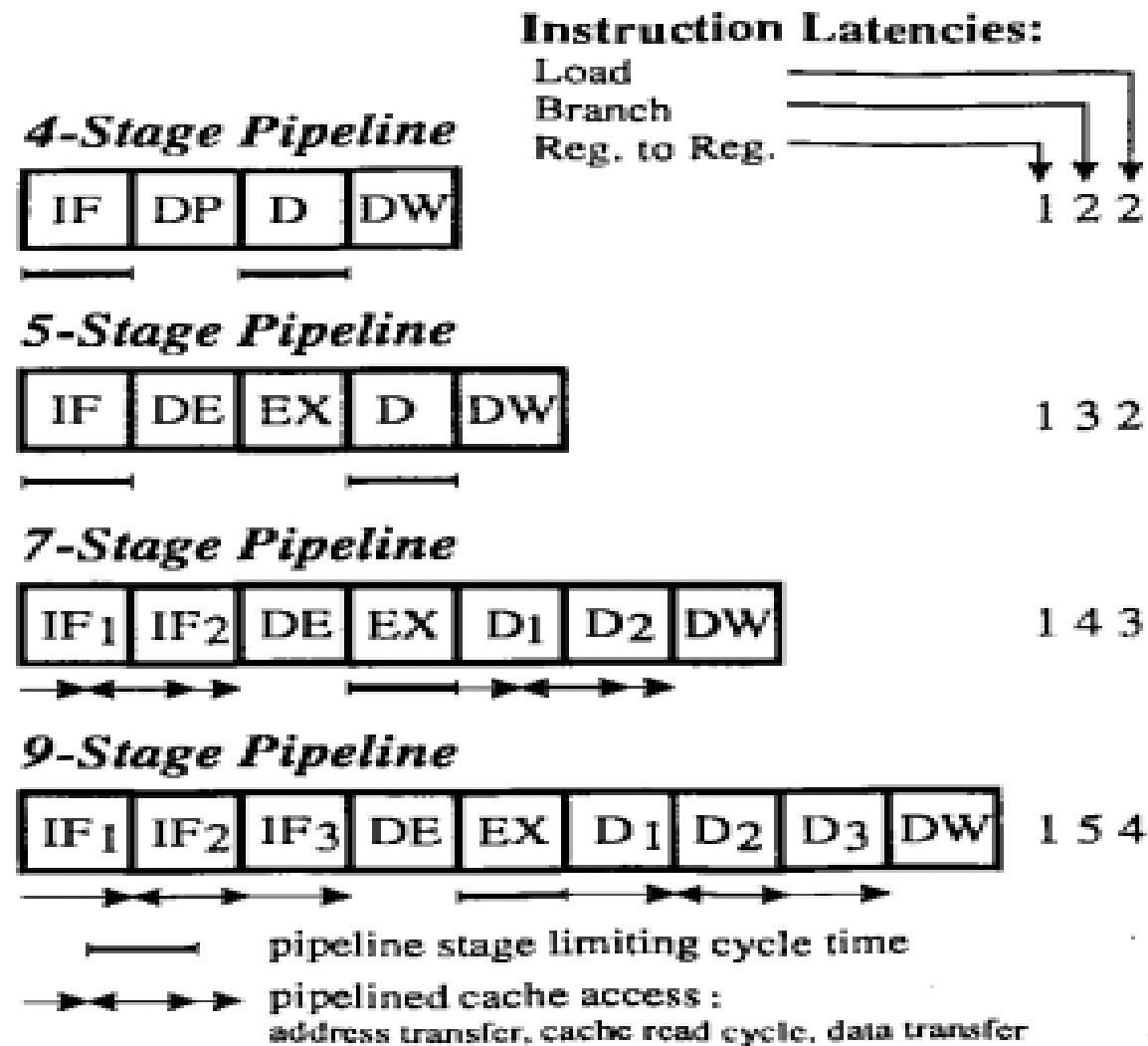


Fig. 2. Potential instruction pipeline schemes for F-RISC/I.

그림 2 는 주기 시간과 CPI를 기준으로 평가

5단계 및 4단계 파이프라인의 주기 시간은 데이터 I/O(D) 단계에서 캐시 메모리 액세스에 의해 설정된 동일한 주기 시간으로 제한됩니다.

7단계 파이프라인은 GaAs 기술에 의해 설정된 주기 시간을 달성할 수 있으며 파이프라인 캐시 메모리 액세스도 제공하지만 주소/데이터 전송을 위한 CPU 사이클은 절반에 불과하다.

9단계 파이프라인은 주소/데이터 전송에 대해 하나의 전체 사이클을 할당하고 캐시 메모리 액세스에 대해 하나의 CPU 사이클을 할당하여 캐시 크기를 늘리고 캐시 누락에 따른 불이익을 최소화하였다.

TABLE I
CPI CONTRIBUTIONS DUE TO BRANCH AND LOAD PENALTIES

<i>Instruction</i>	<i>Instruction frequency</i> ¹	<i>Cost 4-stage</i>	<i>Cost 7-stage</i>	<i>Cost 9-stage</i>
<i>ALU</i>	60%	1	1	1
<i>BRANCH (not taken)</i>	5%	1	1	1
<i>BRANCH (taken)</i> ²	15%	1.368	2.673	3.673
<i>LOAD</i> ³	15%	1.1	1.5	2.3
<i>STORE</i>	5%	1	1	1
<i>CPI_{intrinsic}</i>		1.07	1.326	1.596

¹ Average dynamic instruction mix [8].

² Cost of a taken branch depends upon the branch latency slot fill-in probability of the compiler. The compiler branch fill-in probabilities are taken from [7]. (4-fill = 0, 3-fills = 36.8%, 2-fills = 20.2%, 1-fills=16.5% and 0-fill = 26.5%).

³ Cost of a load instruction depends upon the load latency slot fill-in probability of the compiler. The compiler load latency slot fill-in probabilities are taken from [5].

파이프라인 논문

표 1

일반적인 UNIX 프로그램 집합에서 파생된 동적 명령어 조합에 대한 분기 및 로드 지연 시간의 CPI 기여도

주소/데이터 전송 주기가 길면 MCM에 더 큰 캐시를 구현할 수 있는데, 이는 사용 가능한 주소/데이터 전송 시간 내에 더 많은 SRAM 칩에 도달할 수 있기 때문이다.

그러나 주기 시간이 길수록 (4 단계 파이프 라인) 최대 명령어 실행 속도가 느려집니다.

주소 / 데이터 전송을 위해 더 많은 주기 / 단계 (7 단계 및 9 단계 파이프 라인)를 허용하면 load 및 분기 페널티가 증가한다.

7 단계 및 9 단계 파이프 라인의 1단계 캐시 크기는 주로 열 관리, 네트워크 라우팅 / 토폴로지, 할당된 주소 / 데이터 전송 시간의 세 가지 요소에 따라 달라집니다.

TABLE II
PERFORMANCE VERSUS PIPELINE DEPTH

<i>Pipeline Depth</i>	<i>T_{cycle}</i> (Relative)	<i>CPI_{intrinsic}</i>	<i>CPI_{cache_miss}</i>	<i>Performance</i> (Relative)
<i>4-stage</i>	1	1.07	0.188	1
<i>7-stage</i>	0.625	1.326	0.202	1.317
<i>9-stage</i>	0.625	1.596	0.140	1.159

¹ Cache miss ratios are based on SPEC92 benchmark trace data [11].

파이프라인 논문

표 II는 세 가지 파이프라인 체계의 상대적 성능을 비교.

7 단계 파이프 라인은 4 단계 및 9 단계 파이프 라인보다 성능이 좋다.

9 단계 파이프 라인은 cache miss 페널티가 가장 낮지만 분기 /load 페널티가 크다.

4 단계 파이프 라인은 분기 / load 페널티가 가장 낮고 캐시 누락 페널티가 상당히 적지만 , 주기 시간이 길수록 파이프라인 효율성이 저하된다.

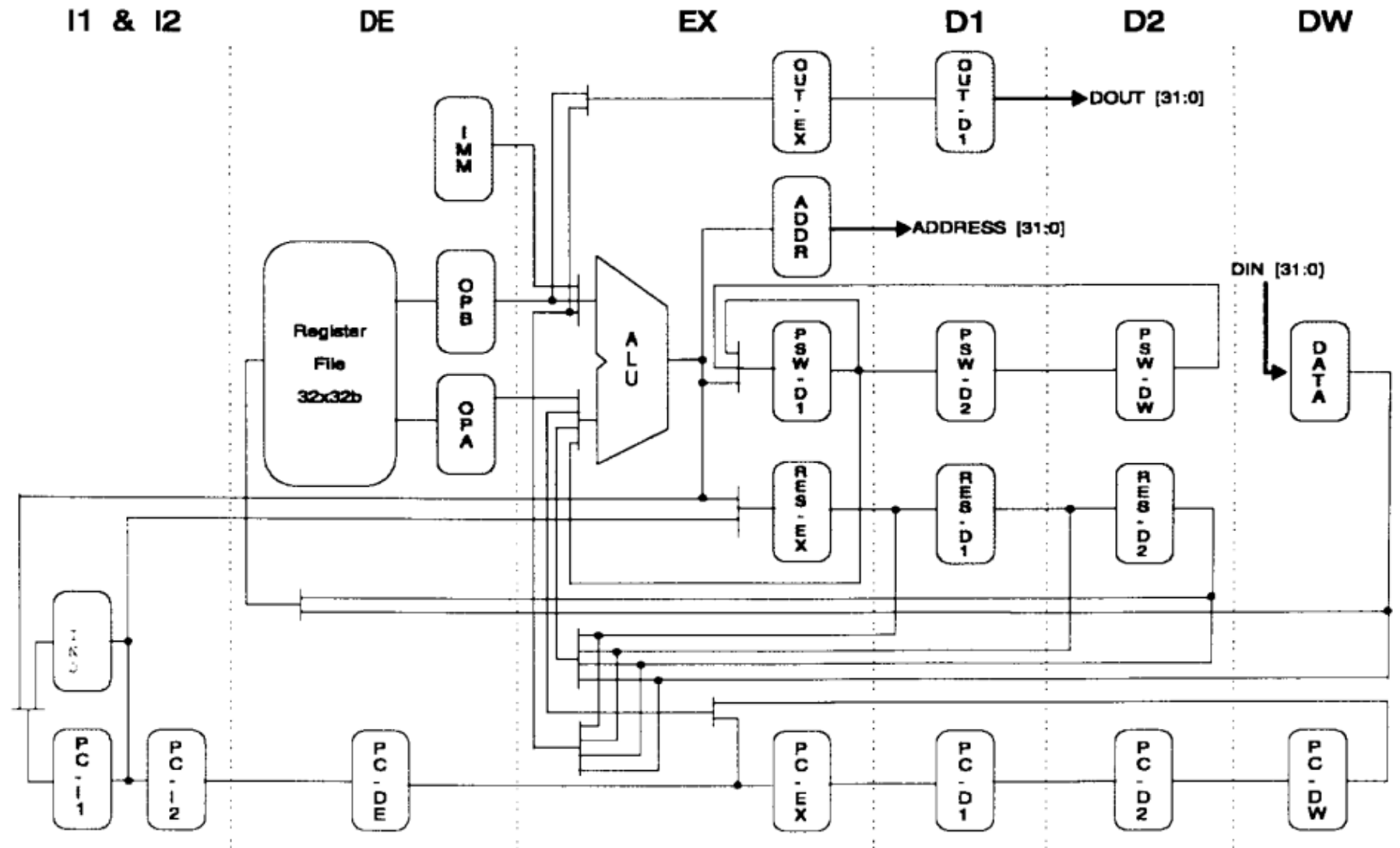


Fig. 7. FRISC/I datapath.

TABLE III
COMPARISONS OF CRITICAL PATH DELAYS

<i>Critical Path</i>	<i>Simulated Delay</i>	<i>Measured Delay</i>
PC increments in I1 stage	4.21 ns	5.0 ns
Register File Read in DE stage	3.12 ns	3.36 ns
ALU Execution in EX stage	5.17 ns	6.0 ns
LSSD latch Delay	1.42 ns	1.48 ns

그림 7은 F-RISC / I의 데이터 경로

중요한 경로는 I1 단계의 PC 증가, DE 단계의 레지스터 파일 읽기, ALU 실행 및 EX 단계의 결과 feed-forward 다.

가장 긴 경로는 결과 레지스터 (RES EX)의 출력에서 시작하여 멀티플렉서와 ALU를 거쳐 RES EX의 입력에서 종료된다..

이 중요한 경로는 ADD 명령이 이전 명령의 결과를 필요로 할 때 사용

표 III 시뮬레이션 결과와 측정 사이의 비교

파이프라인 논문

분기 패널티란?

왜 존재하는가? 위험을 방지하기 위해?

파이프라인을 통해 명령을 얻는 데는 몇 번의 클럭 사이클이 필요하며, 분기 명령은 파이프라인의 중간 지점까지 분기 목표를 결정하지 못할 수 있다. 따라서 대상이 결정되고 다음 명령이 가져올 때까지 파이프라인을 정지해야 한다.

특히 대상 명령이 캐시에 없는 경우(아키텍처에 캐시가 있다고 가정) 이 작업에는 여러 주기가 걸릴 수 있다.)

분기 예측이 정확하지 않으면 ALU가 조건부 분기 또는 간접 분기의 이동 방식을 결정할 때까지 가져오기 작업은 다음에 무엇을 가져올지 알 수 없다.

따라서 ALU에서 분기가 실행될 때까지 대기.

잘못된 예측으로 인해 잘못된 경로에서 가져온/암호 해독된 명령은 무용지물이므로 분기 잘못 예측 패널티라고 합니다.

분기 예측은 일반적인 경우 숨긴다.

또 다른 용어는 "분기 지연 시간".

분기 명령을 가져올 때부터 front-end fetches가 유용한 다음 명령을 가져올 때까지의 사이클 수입니다.

명령이 분기라는 사실은 해독된 후에 알 수 있다.

실행보다 파이프라인에서 더 이전이므로 조건부 또는 간접 분기에 비해 더 작다.

무조건적인 분기에서도 분기 지연 시간이 있다.

파이프라인 논문

로드 사용 페널티란 무엇을 의미?.

로드를 시작할 때부터 데이터를 실제로 사용할 수 있을 때까지의 주기 수입니다
현대의 슈퍼스칼라 비순차 시스템에서는 프로세서가 일반적으로 이 지연 시간을 잘 숨긴다.

메모리에서 레지스터를 로드하는 명령 A가 있을 수 있다.
추가로 피연산자로 레지스터를 사용하는 후속 명령 B가 있을 수 있다.

첫 번째 명령이 완료되는 주기와 두 번째 명령이 완료되는 주기의 최소 간격은 "사용할 부하" 페널티이다.

L1 캐시(대부분)에서 발생하는 메모리 작업의 경우 이 숫자는 약 3 사이클이 될 수 있습니다. L2 히트, L3 히트 또는 메인 메모리의 경우 더 클 수 있다.

캐시 읽기 데이터를 ALU로 직접 우회하므로 메모리 읽기 데이터를 레지스터 파일에 저장한 다음 다시 읽을 필요가 없다.

파이프라인의 슈퍼스칼라(superscalar) 설계에서 가능한 모든 작업을 병렬로 수행한다, 가상 주소를 물리적 주소로 변환하고, 캐시 어레이로 주소를 가져오고, 태그를 비교하고, 패리티를 확인하고, 읽기 데이터를 정렬하고, 결과 데이터를 다시 가져오는 작업에는 정말 엄청난 양의 작업이 있다. ALU로요

파이프라인 논문

일반적인 데이터 위험은 특히 짧은 파이프라인에서 상당히 빠르게 우회할 수 있다

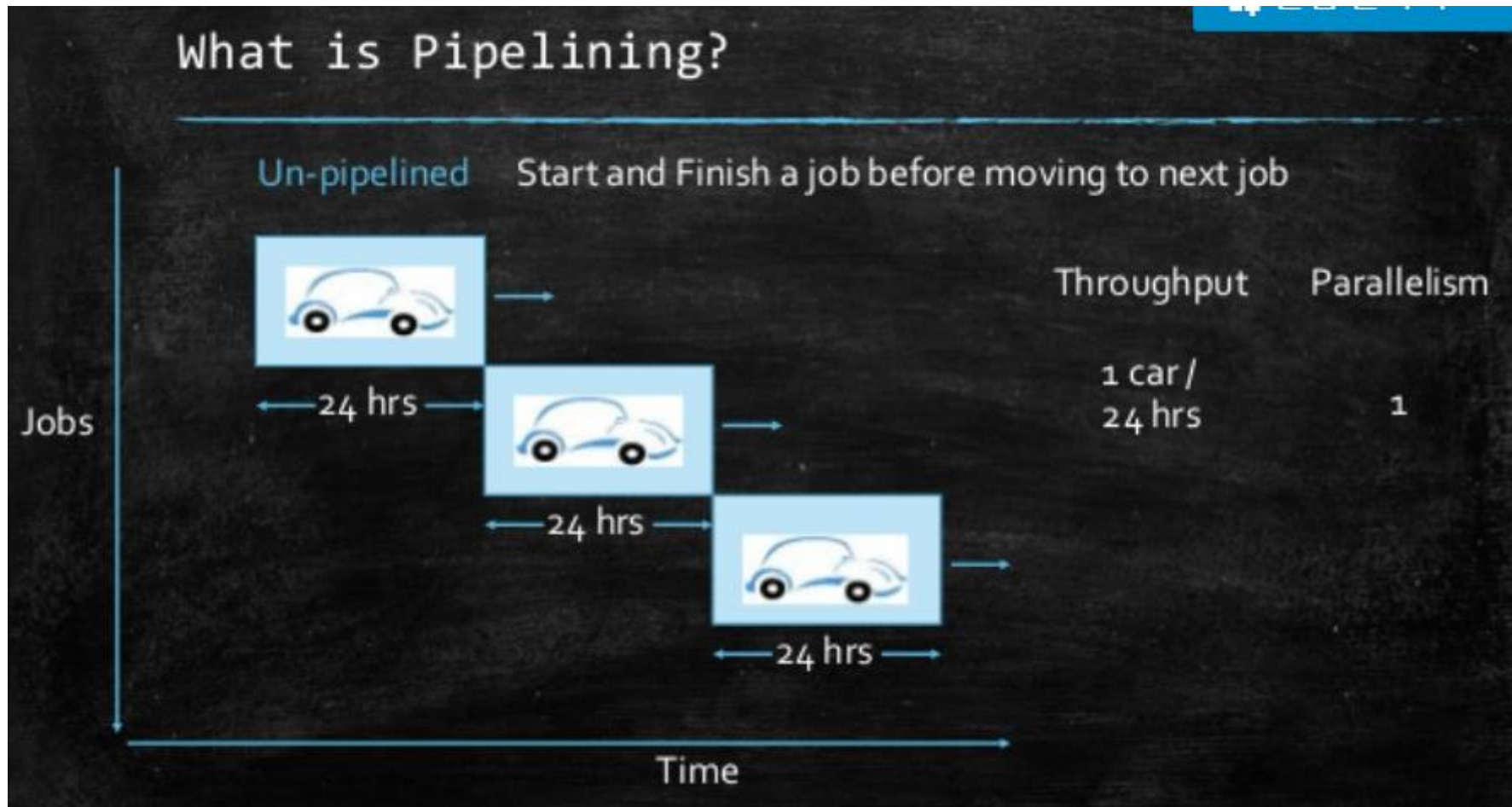
두 번째 명령이 실행되려면

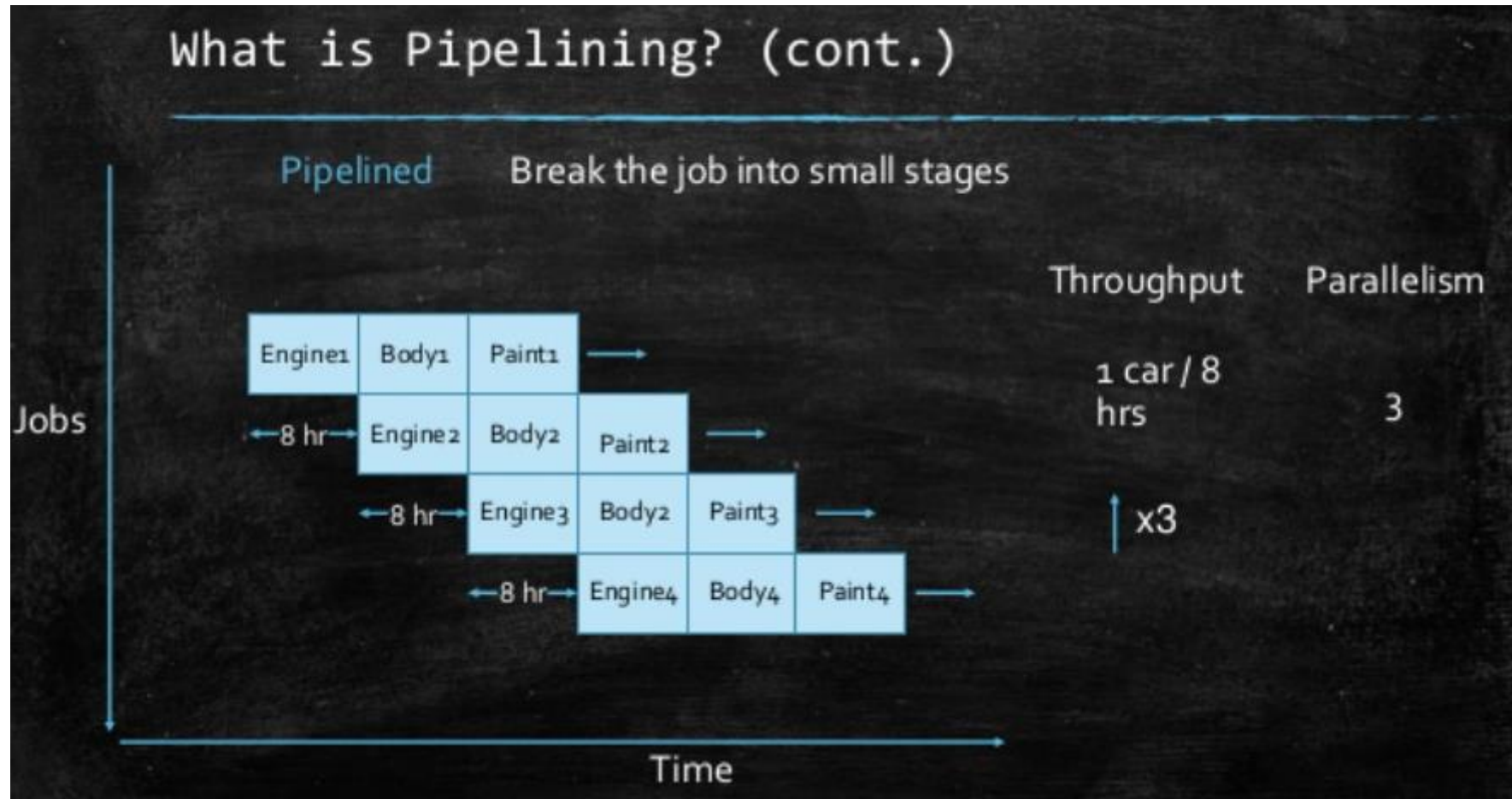
L1 데이터 캐시(일반적으로 여러 사이클의 지연 시간) 또는 RAM/기타 캐시 레벨에서 데이터를 검색해야 합니다.

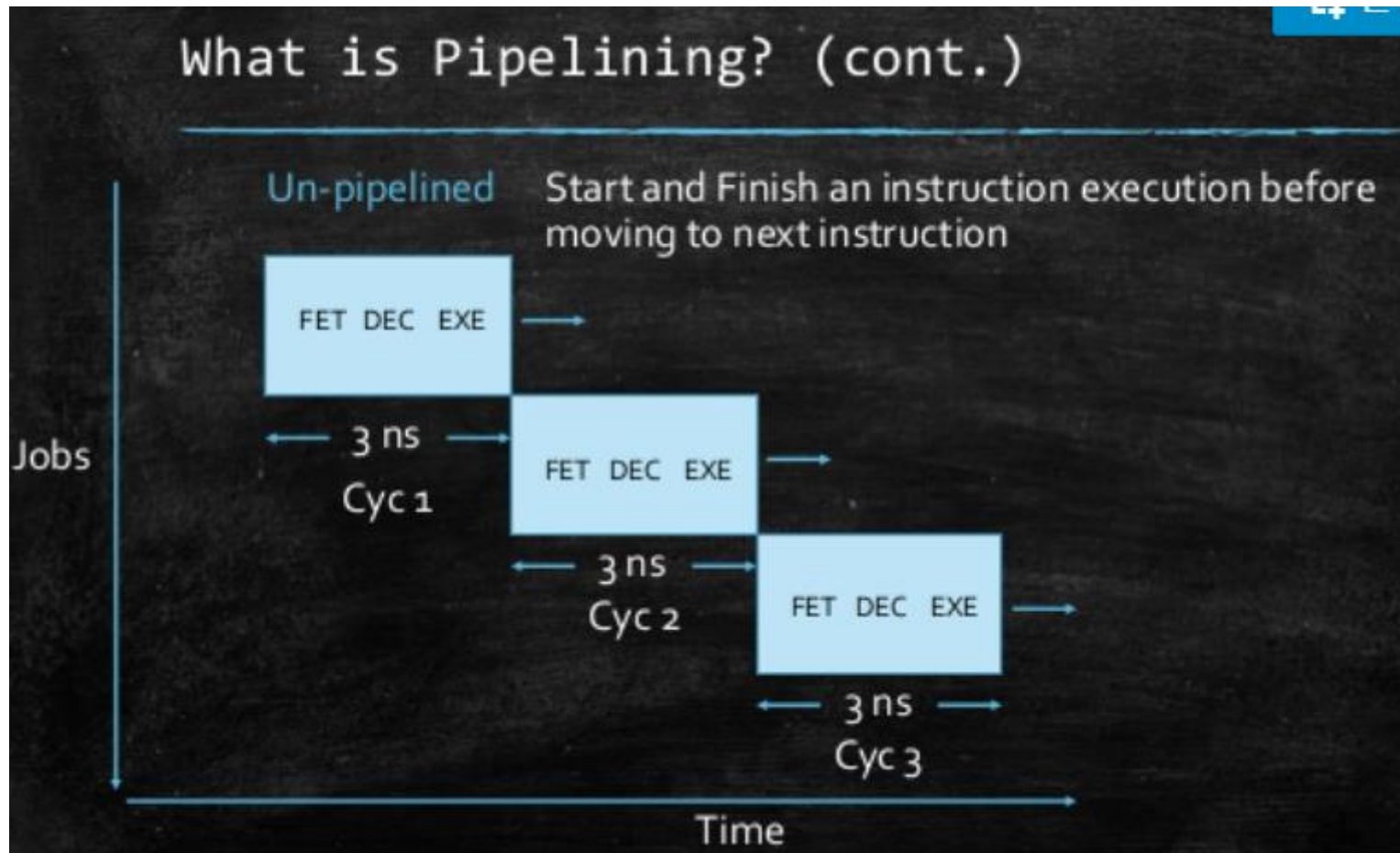
"단순한" 데이터 위험보다 훨씬 더 많은 지연이 있으며, 상당히 변동할 수 있습니다(RAM 접근은 L1 접근보다 약 2배 느림).

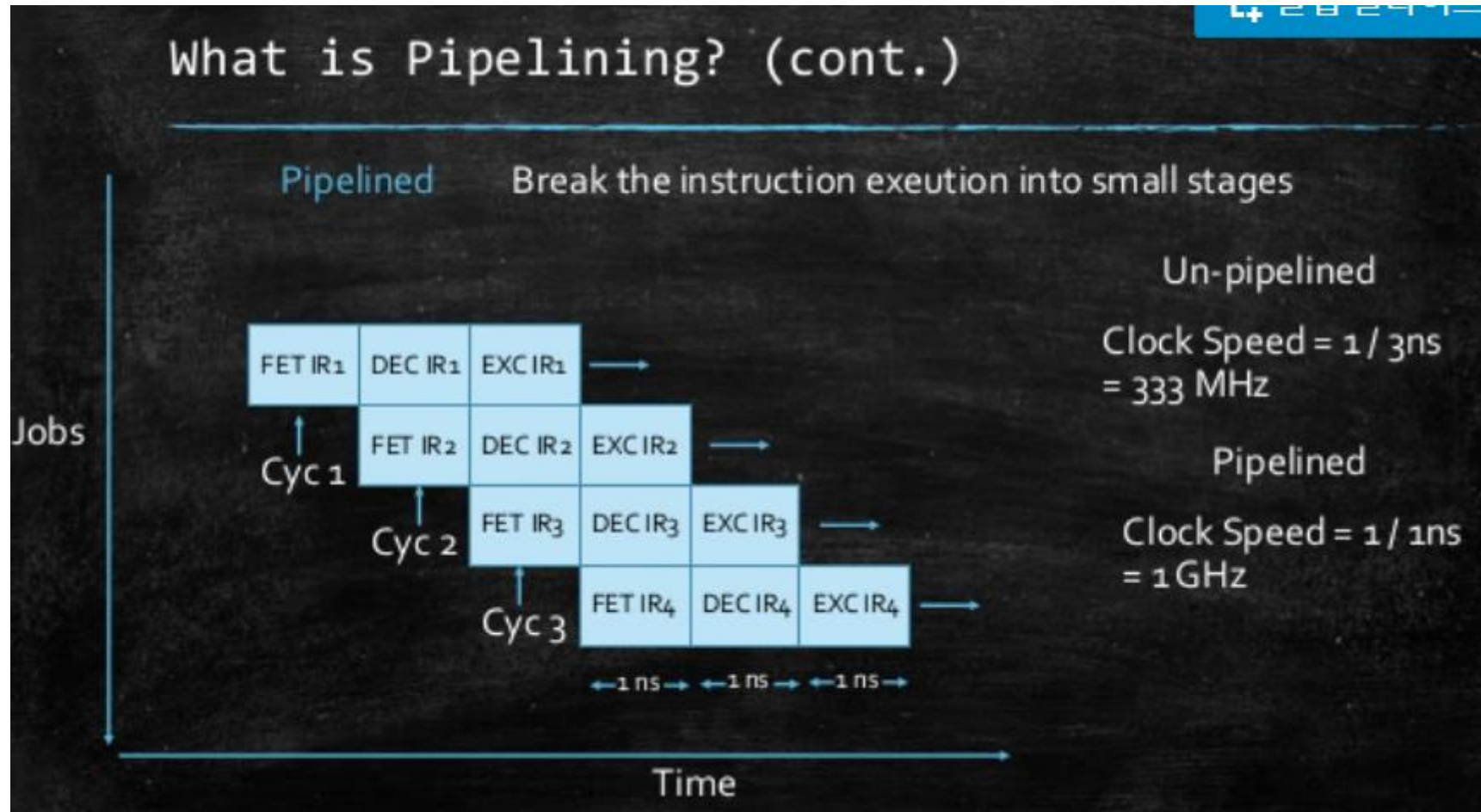
이것이 바로 로드 스토어의 페널티입니다.

로드 명령과 이를 피연산자로 사용하는 후속 명령 사이에 낭비되는 주기는 얼마나 됩니까?









1. 프로젝트 소개

product name

LED 제어

Function

전원부는 5V , 12V , 24V 中 택1

약 5 ~ 10 개 정도의 LED 를 순차적으로 ON/OFF 하며 반복하는 것.

LED 의 ON 시간과 OFF 시간을 버튼을 통해 사용자가 설정.

ON / OFF 시간 및 카운터는 LCD 또는 7-segment 를 통해 숫자를 표기.

카운터 설정을 통해 반복되는 구간을 설정.

LED 구간을 정해 반복 작동하기 위한 값으로 카운터가 있다.

- 5 로 설정하면 1 ~ 5 라인만 반복.
- 10 으로 설정하면 1 ~ 10 라인만 반복

2. 프로젝트 일정 및 계획

부품 구매 : 판다파츠 또는 디바이스 마트에서 부품 구매.

회로 설계 : 오실로스코프 , 함수 발생기 , 디지털 멀티미터 2개 사용.

PCB 설계 : 만능기판으로 납땜.

펌웨어 작성 : Function 을 C 코드로 구현.

동작 확인 : 장시간 동안 동작해도 문제 없는지 확인. (디버깅)

Week	1	2	3	4	5	6	7	8
부품 구매 회로 설계								
PCB 설계								
펌웨어 작성								
동작 확인								

보류

통신 I/F 회로 : interface 의 약어 , 줄여서 I/F 로 사용

유튜브 : (Korean) 새로운 MPLAB PICkit 4 인-서킷 디버거로 개발 시작하기

네이버 (선진 유통) : [정품 100% 보장] PICKIT4 PG164140 / 마이크로칩 개발장비

ATmega4809 PF (PDIP 40 pin)

End of Document