1. Computer의 카테고리

1) Desktop (=PC =Workstation)
- IBM-PC
- Apple

2) Server
- reliability (;dependability), availability, expandability

3) Notebook

4) Embedded computer
- real-time(계산하는 시간이 응용 목적에 아무 지장 없음.)
- SoC(System on Chip) : CPU core + custom hardware. 소형화, 가격down.

5) Smart phone

2. Com의 function

1) Data processing
2) Data storage
3) Data movement
4) Control

3. com의 역사

1) Eniac
   - programmed manually by switches
2) 국가슈퍼컴퓨터5호
   - petaFLOP : floating point number operation per second. 실수 계산 얼마나 잘하는지 컴퓨터 성능. 실수(floating point; 부동소수점 숫자)는 정수(integer; fixed point num)보다 작업 20배 어려움. peta는 천 조.
   - Core : cpu랑 똑같은 말임. 칩 하나에 core 여러 개 넣음. 칩 하나가 cpu인데 그 안에 core 여러 개 있음.

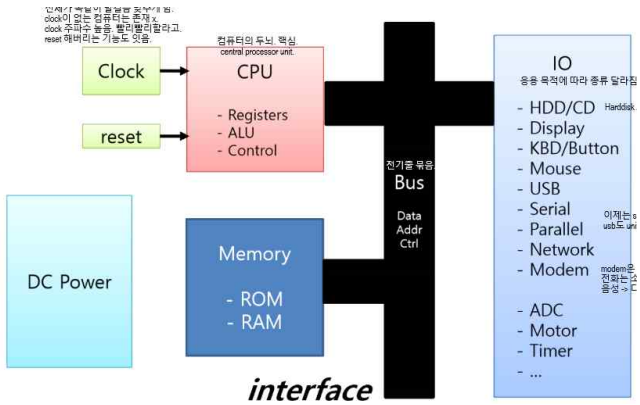
4. com의 여러 가지 부품

1) power supply. 전원 장치. dc.
2) 메인보드. 주기판. 마더보드
3) cpu. 반도체 칩.
4) 보조칩 LGA775. 다른 칩들과 연결해줌.
5) 메모리. 램 끼울 수 있는 슬롯.
6) rear panel.
7) pcie16. 외부 장치와 연결하는 아주 큰 슬롯.
8) PCI. 하드웨어 만든거 연결.

5. Intels의 첫 process, 4004.

1) 740kHz의 clock
2) 2300 transistor
3) 16 pins
4) 12 bit address
5) 8 bit instructions
6) 4 bit data words == 4 bit 컴퓨터임.

6. Computer의 components



1) CPU
중앙연산장치. central process unit. (Register, ALU, Control). Clock과 reset.

2) Memory (ROM, RAM).
- Internal(on-chip; cpu칩 기준 그 안에 있는 거) & external
- ROM(PROM, EPROM) (read only memory. 이미 새겨놓은 거. 모든 컴에 있. power 키면 구동됨.) & RAM (read write memory)
- DRAM(dynamic) & SRAM(static)
- Cache, Main memory, Virtual memory

3) IO
- Serial, Parallal, USB(universal serial bus)
- Graphic display
- Storage (HDD, CD)
- KBD (button), Mouse, ..
- ADC, DAC

4) BUS (Data, Addr, Ctrl)

5) DC Power
- 5, 3.3, 9, 12, 18V 등.

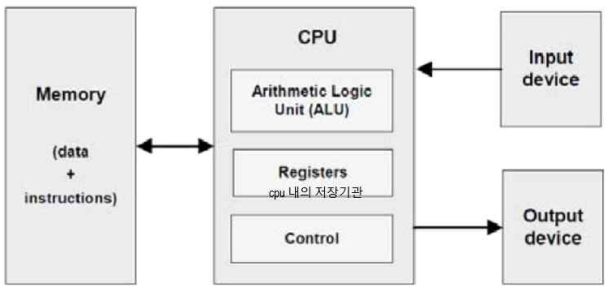
6) Interface : bus에 장치 붙이는 행위

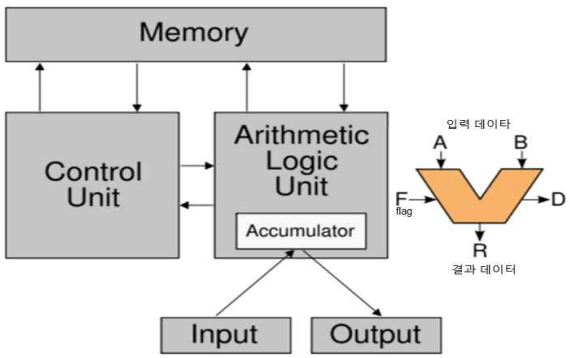
7. Computer 설계하는 일. Architecture.

1) 컴퓨터 설계자는 computer architect.
   사용자는 programmer.
2) Register data width (8/16/32/64) (n bit computer인지 결정.) (n 부품 처리기) (하드웨어)
3) Instruction set
4) Addressing modes and methods
5) Memory controllers and hierarchies
6) System interconnects (bus 등)
7) Multi-processing (cpu 여러 개)

8. Von neumann machine

1) 메모리 한 덩어리에 data와 instruction(program) 넣고 CPU에서 하나씩 꺼내 쓰자. 즉, program을 메모리에 넣자.





2) Havard architecture : PM과 DM을 분리함. pipelining 할려고 하바드 구조 도입한 것임. instruction과 data를 동시에 읽으려고!

9. CPU must

1) Fetch instruction.
2) Interpret(decode) instructions.
3) Fetch data. (register로)
4) Process data.
5) Write data.

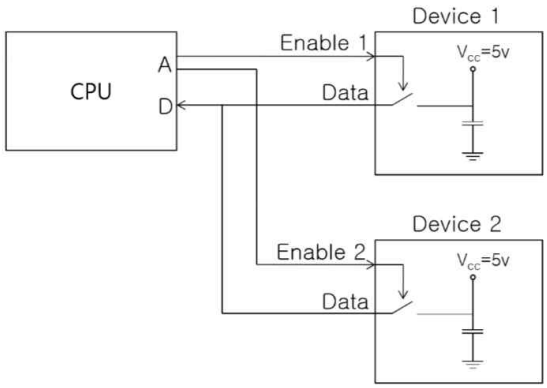
10. CPU가 mem이나 IO에 Access하는 법

1) Read or Write

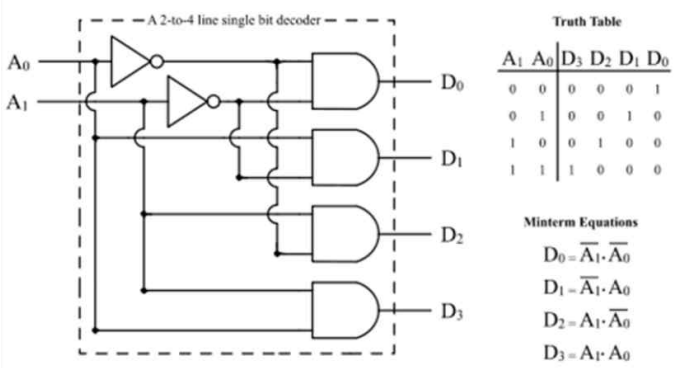
- cpu가 주체. 외부의 데이터를 cpu내부로 가져옴.
- 원하는 때에만 가져오고 싶으니까 address A라는 control 신호 있음.
- Device는 memory 혹은 IO.
- enable or disable(open state)됨.
- enable 되면 data선을 통해 cpu로 전달됨.

2) Addressing

- enable 선이 어느 device의 data 들고오게 할건지 통제함. decoding함. 암호화함.
- n개 address line => 2^n개의 enable lines.
- 32 address line => 4G enable lines
- 64(48) address line => 16 exa (256T)

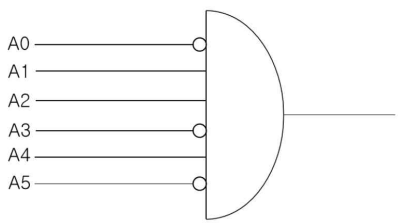


3) Address Decoder (2 to 4)



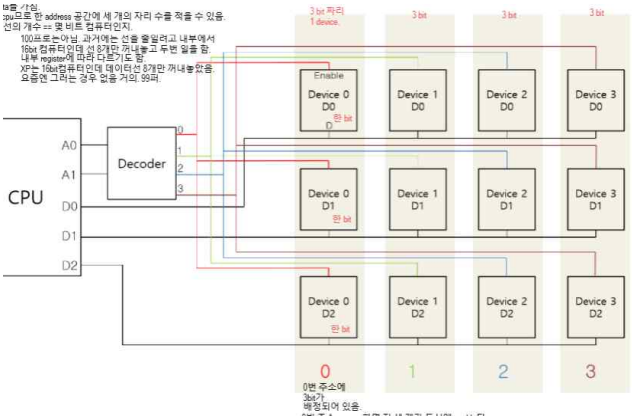
4) Decoding Example

- Address space :6bits (0x00~0x3F)
- Given address : 0x16 => 0b 01 0110



11. BUS Interface

(4device X 3bit = 1MEM) = 총12bit



1) 2 Address line (A0, A1)

2 address bus 폭.

2^2 = 4 Address space.

총 0~3번 주소 있음.

00 ~ 11.

4 device까지 가능.

4 maximum memory capacity.

2) 1MEM 당 4 device

n to 4 decoder, 즉 n=2.

2 address line 중

2개를 디코더에, 0개를 /CS에 씀.

3) 3 data line (D0, D1, D2)

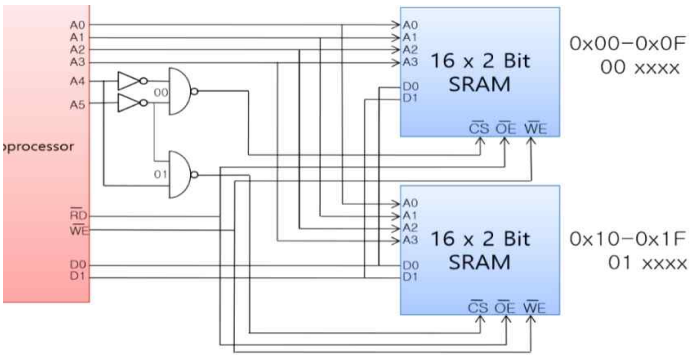
3 bit computer	3 bit cpu
3 bit width register	3 bit width data bus
ALU의 bit수 3임.	3 bit data words.
3 bit를 한번에 계산.	device 당 3 bit data
한 instruction이 ALU에 의해 process되는 bit수 3	한 주소 access=> 3개 동시에 enable됨.

※ 모든 device에는 enable 선 연결돼있음.

- AVR은 8bit computer
- 내pc는 64bit computer
- 인간은 1digit computer

23. Memory Interface

2개 X (16device X 2bit = 1SRAM) = 총64bit



1) 6 Address line (A0~A5)

6 address bus 폭.

2^6 = 64 address space.

총 00번 ~ 63번 주소 있음.

0x00~0x3F, 0000 0000 ~ 0001 1111.

64 device까지 가능.

6 maximum memory capacity.

2) 1SRAM 당 16 device

n to 16 decoder 사용. 즉, n=4.

6 address line 중

4개를 디코더에, 2개를 /CS에 씀.

sram①은 00번~15번 주소 받아야함.

0x00 ~ 0x0F. 0000 0000 ~ 0000 1111.

sram②는 16번~31번 주소 받아야함.

0x10 ~ 0x1F. 0001 0000 ~ 0001 1111.

쉬는 공간 32번~63번.

0x20 ~ 0x3F. 0010 0000 ~ 0011 1111.

A0~A3의 4 address line으로

주소의 뒷 4자리 결정. ____ ****.

A4, A5의 2 address line으로

어떤 sram 구동할지 결정. __** ____.

/CS로 연결함. chip selector, chip enable.

/CS가 0이여야 그 sram 활성화.

sram①은 A5, A4가 00이면 구동.

sram②는 A5, A4가 01이면 구동.

빈공간 은 A5, A4가 1?이면 구동.

3) 2 data line (D0, D1)

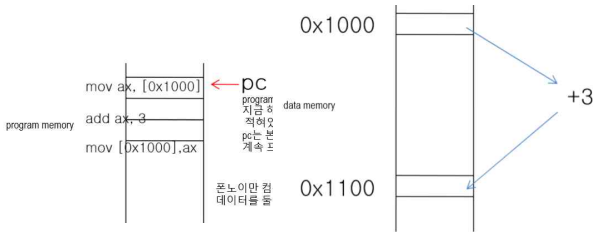
2 bit computer	2 bit cpu
2 bit width register	2 bit width data bus
ALU의 bit수 2임.	2 bit data words.
2 bit를 한번에 계산.	device 당 2 bit data
한 instruction이 ALU에 의해 process되는 bit수 2	한 주소 access=> 2개 동시에 enable됨.

4) control line /WE

write enable, active low. 0이 true.

12. execution (x86) 예

0) 그림



- pc는 program counter라는 특별한 register. 지금 해야 할 instruction이 메모리 몇 번지에 있는지 주소가 적혀있음. 폰노이만 com에 pc 반드시 있어야 함.

1) C language로 1000번지에 있는 내용에 3을 더해서 1100번지에 쓰자.

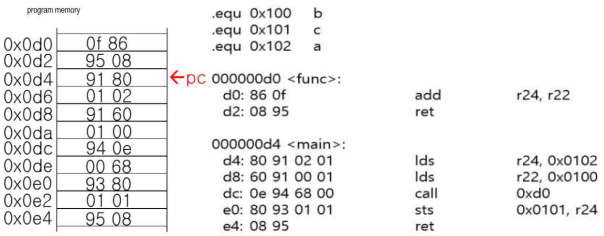
```
short a, *ptr;
ptr = 0x1000;
a = *ptr+3;
ptr = 0x1100;
*ptr = a;
```

2) x86 assembly. ax는 register임.

```
mov ax, [0x1000]
add ax,3
mov [0x1100], ax
```

13. execution (AVR) 예

0) 그림



2) AVR instructions

1) C언어

```
char a,b,c;
char func (char x, char y) { return (x+y); }
void main() { c = func(a,b); }
```

14. Instruction Subcycles and Pipelining

1) 한 개의 instruction 쓰기 위한 과정

① IF : Fetch instruction from memory

② ID : Read registers while decoding instruction.

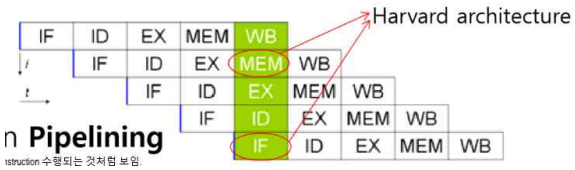
③ EX : Execute the operation or calculate a memory address

④ MEM : Access an operand in data memory

⑤ WB : Write the result into a register

- 5 clock이 있어야 한 instruction 수행 가능.
- clock이 1MHz면 1초에 20만 개의 instruction 수행 가능.
- clock은 crystal 진동시켜서 0,1을 규칙적로.
- 1MHz면 1초에 백만번 진동. 1cycle은 1us.
- 1GHz면 1초에 10억번 진동. 1cycle은 1ns.

2) Pipelining



- "1 instruction / cycle "
- Single clock cycle execution.
- clock이 1MHz인 건 불변.

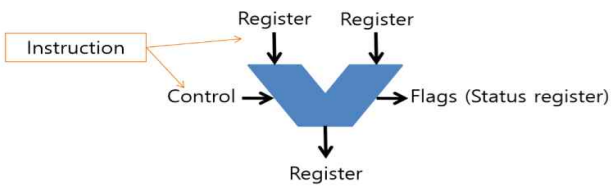
15. CPU 안 부품 3가지

0) core

- cpu 안 cpu를 cored cpu라 함.

1) Functional Units

- ALU = Arithmetic and logic unit
- integer arithmetic operation, bitwise logic operation, bit-shifting operations.
- 모든 컴퓨터는 ALU 하나 이상 있어야 함.
- FPU (floating point unit). 실수 연산 빠르게 助



2) Registers

- operands와 계산된 결과를 저장.

(1) General purpose register; accumulator

- 32 x 8 register (R0~R31)의 AVR
- 6 x 32 register의 Pentium
- 14 x 64 register의 i7.

(2) Program Counter; instruction pointer

- 폰노이만 컴에는 무조건 있음.

(3) Status register; flag register;

- AVR에선 SREG;
- conditional branch instruction 위해 반드시 필요. (; 조건에 따라 건너뛰는 거)

⓪ C; carry; 숫자 넘칠 때. 빼기할 때 borrow로서도 사용됨.
① Z; zero; 빼서 같은지 파악.
② N; negative; 대소비교.
③ V; 2's complement overflow; 8bit 넘치는지.
④ S; sign; N xor V; 대소비교 음수.
⑤ H; half carry;
⑥ T; bit copy storage;
⑦ I; global interrupt enable

(4) Stack pointer

① FIFO : 문두개버스.
- 소프트웨어로, 포인터가 2개인셈.

② LIFO : 문한개버스.
- buffer 많이 씀. 특정 area를 잡아놓고 창고처럼 쓰는 것.
- ram 중에 일부 영역 할당해서 사용함.
- return address (call, ret)
- save/restore registers (push하면 데이터 넣고 sp한칸 올려라, pop하면 데이터 빼고 sp한칸 내려라)
- parameter도 passing함.

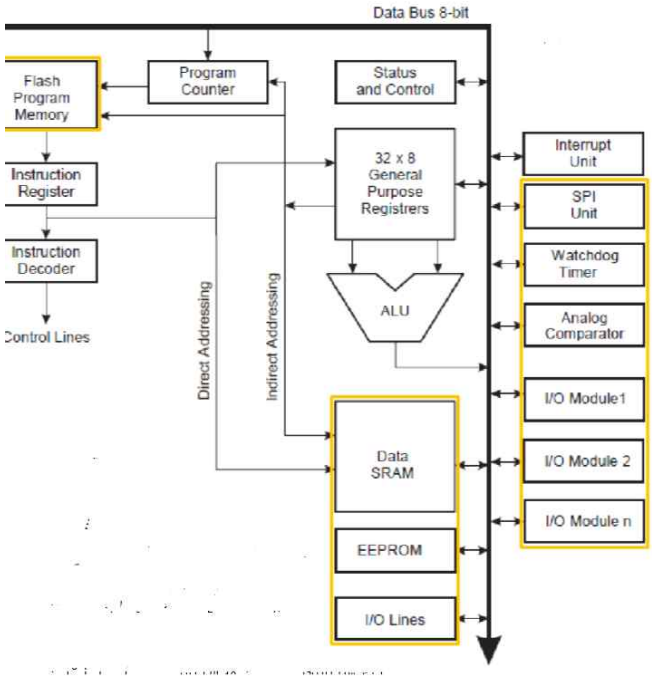
(5) HW control register

3) Control Unit
- Program Counter (PC)
- PC위치의 memory로부터 instruction 읽음.
- PC가 가르키는 번지수에서 instruction register, decoder가 수행.
- instruction을 interpret함.
- 다른 components에게 뭐 해야되는지 signal줌.

4) Word size
- 한 instruction에 ALU 의해 process되는 bit 수.
- register의 width

16. AVR의 CPU

1) Advanced RISC architecture
2) Clock : 8/16 MHz
3) Havard architecture : PM(16), DM(8)
4) 133 powerful instructions. Most single clock cycle execution
5) 32 x 8 General Purpose Working Registers
; 범용 계산용 register가 32개 있다.
; 8bit computer니까 한 레지는 8bit.
+ Peripheral Control Resgisters
; 주변 장치 control 할 레지스터.
6) Up to 16 MIPS thoughput at 16MHz.
; 클락이 16MHz면 최대 16mips다.
; throughput은 처리 속도.
7) On-chip 2-cycle Multiplier



17. Instructions

1) 의미 : fundamental unit of work

2) 구성

(1) Opcode; 동사; operation to be performed; ADD
(2) Operands; 목적어; data/locations to be used for operation; R0, 3

3) 예시

: ADD R0, 3 : R0라는 register에다가 3이라는 덧셈을 해라.

4) decoded, 즉 해석되는 법
- instruction은 encoded된다, as a sequence of bits (jst like data)
- sometimes have a fixed length (16or32bits)
- control unit이 operation을 interpret 한다.
- control signal이 operation을 carry out함.

5) ISA; instruction set architecture

- cpu 디자이너는 instruction set을 설계해야함.
- 결국 isa가 컴퓨터 구조 결정하는 셈.
- AVR은 130개 정도 됨.

(1) Arithmetic instructions; add, subtract

(2) Logic instructions; and, or, xor, not

(3) Data instructions; move, input, output, load, store

(4) Control flow instruction; goto, if, call, return; 프로그램 흐름 제어. branch instruction.

- branch instruction 때문에 status register 필요한 것이다. flag 필요해서.

18. CISC와 RISC

1) CISC

- complex instruction set computer.
- 하나의 instruction이 일을 많이 하도록, 즉 microcode 여러 개 있도록 강력하게 만듦.
- memory 아낌.
- 여러cycle 당 1instruction.
- 범용register 낮춤. (Pentium:8)
- ex) x86, 8051

2) RISC

- reduced instruction set computer
- 1cycle 당 1instuction.
- 간단히, 개수 줄임.
- 컴퓨터 성능 높아짐.
- clock 높아짐.
- 범용 register 큼. (AVR:32)
- instruction set이 limited & simple.
- instruction pipeline 강화.
- compile time/effort 많음.
- ex) AVR, ARM, SPARC(Sun)

19. Numbers

0) only binary numbers(0,1), no minus.

1) Integer

- fixed point
- bits(8/16/32)따라 byte/shortword/longword
- singed/unsigned 표현 가능.

(1) 1's complement

: bitwise not한 것.

1은 0000 0001

-1은 1111 1110

0은 0000 0000

-0은 1111 1111 ?

(2) 2's complement

: 음수는 1's complement +1 한 것.

127은 0111 1111

1은 0000 0001

0은 0000 0000

-1은 1111 1111

-2는 1111 1110

-128은 1000 0000

8 bit 경우 -128~127 표현 가능.

8->16 bit extension은

signed면 부호 따라 붙이면 됨.

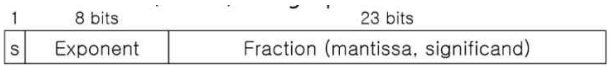
-1면, 1111 1111 => 1111 1111 1111 1111

unsigned면 0 붙이면 됨.

255면, 1111 1111 => 0000 0000 1111 1111

2) Floating point

$$N=(-1)^s \times 1.\text{fraction} \times 2^{(\text{exponent}-127)}$$



3) Character

(1) ASCII code : 1byte.

'A' = 0x41임. 'A'='B'-1.

(2) 한글 : 2byte. 완성형, 조합형.

(3) Unicode : 2byte.

20. Endianness

- byte addressable CPU

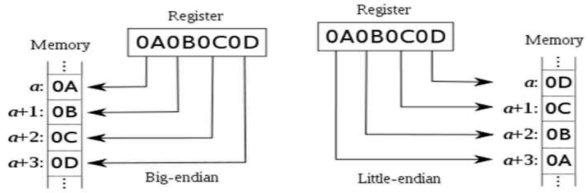
1) MSB, LSB

- most/least significant bit

2) Big endian, Little endian

(1) Big endian : big end first ex) powerPC

(2) Little endian : little end first ex) x86 (PC)



21. Pointer

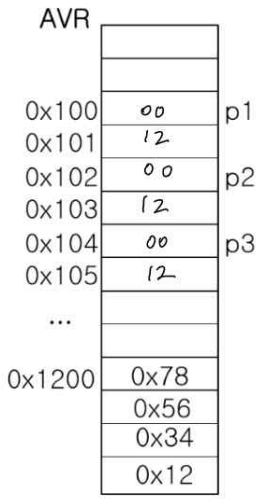
1) 컴퓨터 별 address 크기

(1) AVR은 16bit address. 한 p는 2byte.

(2) 윈도우32는 32bit address. 한 p는 4byte.

(3) 윈도우64는 64bit address. 한 p는 8byte.

2) 예제



만약 주소0x1200에 0x78 0x56 0x34 0x12가 살고 있더라면,
char *p1, short *p2, long *p3;
p1=0x1200;
p2=0x1200;
p3=0x1200; 하면
*p1은 0x78, *p2는 5678, *p3은 0x12345678.
&p1은 0x0100, &p2는 0x0102, &p3는 0x0104.

3) 예제

Memory 0x1000번지에 0xff, 0x2000번지에 100을 써라.

```
char *p;
p=(char*)0x1000;
*p=0xff;
p=(char*)0x2000;
*p=100;
```

4) 예제

*p=0xff; 하면 전구 불 다 킴.
*p=0x0; 하면 전구 불 다 끔.
*p=0x0f; 하면 왼4전구 끔, 오4전구 킴.

5) 예제

Memory 0x2000번지에 0x3000번지의 값을 더하라.

```
char *p1, *p2;
p1=(char *)0x2000;
p2=(char *)0x3000;
*p1=*p1+*p2;
```

6) 예제

```
unsigned short *ptr;
unsigned long x,y;
x=0x12345678;
ptr=(unsigned short *)&x;
y=*ptr;
```

하면
y는 0x5678이 됨. 그리고 little endian이니까 78 56 00 00 순으로 바이트 들어가 있겠지.

7) 예제

```
char a[4] = {0x12, 0x34, 0x56, 0x78};
long x, *ptr;
ptr = (long *)a;
x=*ptr;
```

하면
x는 0x78563412이고, little endian이니까 메모리엔 0x12, 0x34, 0x56, 0x78 순으로 들어가 있을 것임.

22. BUS

1) 3종류의 버스 줄

(1) data line

- data bus의 폭 = 몇 비트 컴퓨터 = register 비트

(2) address line

- cpu(master)는 만들고, mem,io(slave)는 받음.
- address bus의 폭 = maximum memory capacity of system.
- ex) pentium은 32bits (4G address space)
- ex) avr은 16bits (64K address space)
- ex) 내pc는 64bits.

(3) control line

- memory read/write signal.
  ex) avr에선 /RD, /WR, ALE 있음.
  슬래쉬는 active low를 의미.
  0이 low, true, active. 더 안정.
- interrupt request
  ex) avr에선 INT0:7 있음.
- clock signal

(4) +) Power, GND line

2) 데이터 전송 방법

(1) 직렬 : usb, ram, 기지국.

(2) 병렬 : hdd, 컴퓨터, avr.

24. BUS arbitration

1) 목적 : cpu(;core)가 여러 개인 컴퓨터 (cpu1, cpu2, ..., DMA controller), 즉 master가 여러 개면 bus 통제자 필요. memory, io는 slave. 데이터 간 충돌 막기 위해 필요.

2) 종류

(1) centralized

중앙집권형. 별도의 control 장치 있다. bus controller(=arbiter)라는 HW가 별도로 붙어 있음. cpu는 개인의 통제 하에 master역할 함.

(2) distrubuted.

권력분산형. 각 cpu마다 그런 장치가 있음. 누가 쓰고 있으면 기다림.

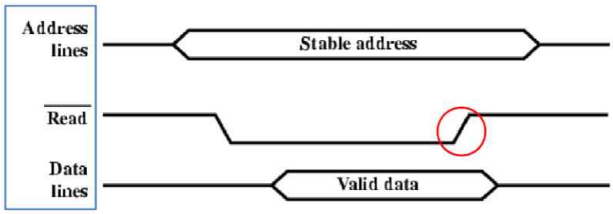
3) PCI

peripheral component interconnection. 버스 arbitration이 있는 bus임. pc에 HW 끼우려면 어려운데, 그래픽 카드도 pci에 꽂혀있음.

4) DMA

direct memory access. HDD에서 대량의 data 읽어들이려 하는데, cpu는 1word밖에 처리 못해서 느리니까 dma는 cpu의 명령을 받아서 데이터 loop으로 불러오는 역할만 함. 대량의 메모리 전송 가능. dma controller도 일종의 bus master.

25. Asynchronous Timing; Read diagram



- 비동기 방식. read가 rising edge일 때 데이터가 valid하다. 신호를 가져가겠다.
- 읽는 기간 동안, 읽으라는 명령어 있으면 /read가 fall 하고, 실제로 읽어야 할 때 rise 한다.