\*필수개념\*

GB = 10^9 , 이진법에서는 2^30

MB = 2^6, 이진법에서는 2^20

1GHZ = 1초에 10^9 번 반복

Million: 1000000= 10^6

10진수 2진수로 바꾸는 방법(정수) : 1이 나올 때까지 10진수를 2로 나눔

2진수 10진수로 바꾸는 방법(정수) : 128(음수 가능) 64 32 16 8 4 2 1

10진수 2진수로 바꾸는 방법(실수) : (IEEE 32bit 실수표현)

1. -1.5 -> 음수 이므로 sign bit는 1, 실수부 1.5를 이진법으로 바꾸면 정수부 1은 내비두고 0.5를 0.5 x 2 =1 이므로 1.1 이 이진법 표현이다. 이를 다시 실수표현방법 +- 1.M x 2^n 꼴로 바꾸어 주면, -1.1 x 2^0이다.

0+127=127이므로 127은 01111111 이고 그 아래 bit는 0으로 채워준다

1. -1/32 = 0.03125 이고 0.03125를 이진법으로 바꿔주면

0.03125 x 2 = 0.0625 - 0

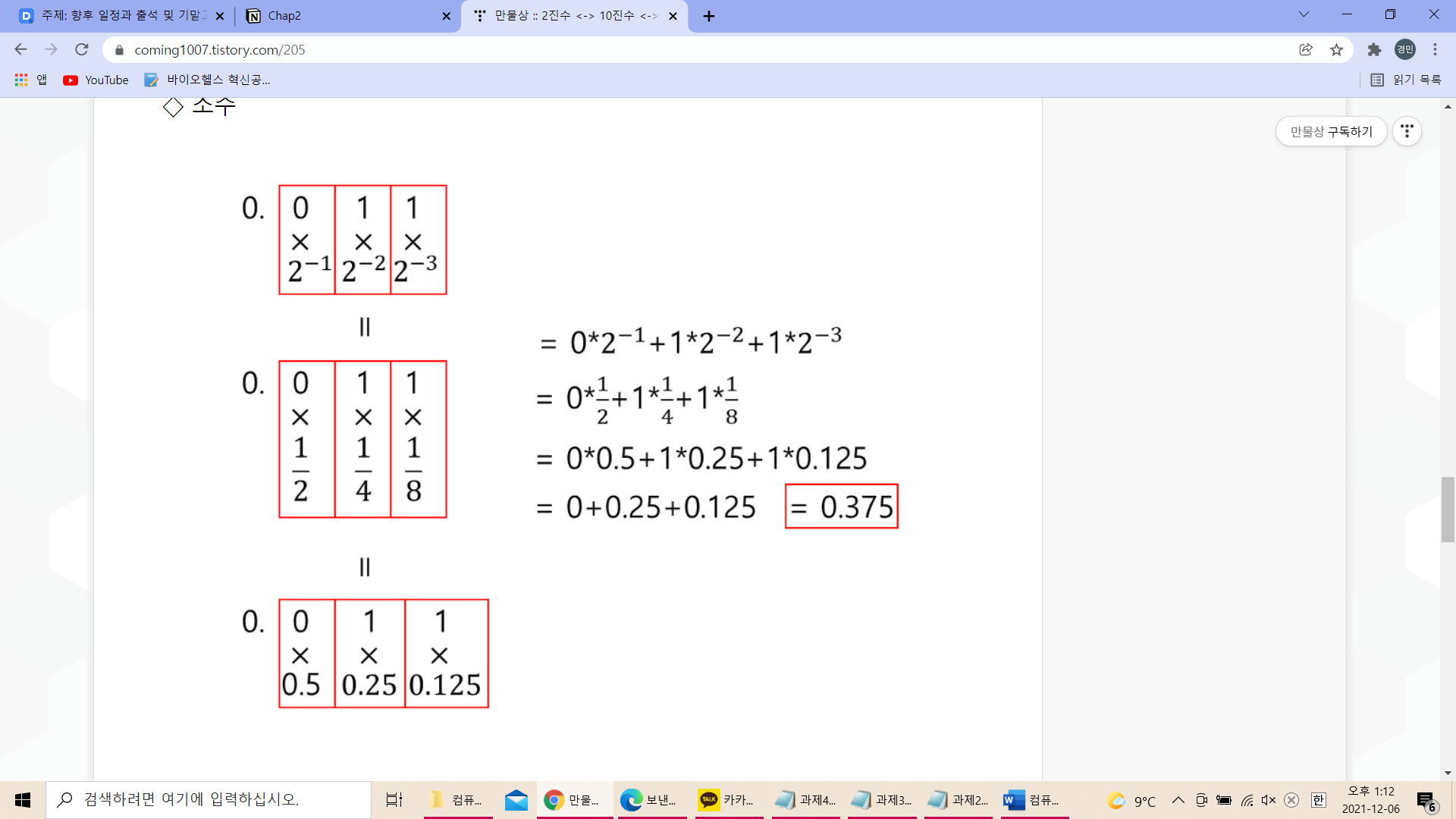
0.0625 x 2 = 0.125 - 0

0.125 x 2 = 0.25 - 0

0.25 x 2 = 0.5 – 0

0.5 x 2 = 1.0 – 1 이므로 0.0001 이다.

2진수 10진수로 바꾸는 방법(실수) :



Chap 2. 성능 이슈

성능을 높이기 위한 디자인

Microprocessor 속도 up 위한 기술

* 1) pipeline 2) 분기예측 3) data flow 분석 4) speculative extection

Microprocessor와 같은 CPU의 성능을 메모리가 따라가지 못하는 문제 해결

* 1) 느린 메모리 접근 시간 줄이고 캐시 사용 2) MM에서 한 번에 많은 양 읽어들임

3) cpu, mm 연결하는 system bus 속도 up(계층적으로 만듬)

4) DRAM 구조 변화시켜 빠르게 MM 가져오기(mm자체 성능 up)

Chip의 구조와 조직 발전

* 1) 18개월 마다 더 많은 트랜지스터를 IC chip에 넣을 수 있음

2) 남는 부분엔 캐시 더 넣음

3) gate는 더 촘촘해지고 data 주고 받는 시간 빨라짐

4) 멀티코어

Multicore(멀티코어): 하나의 프로세서 칩에 여러 개의 core 넣은 것

1. 복잡한 core 한 개보다는 단순하고 간단한 core 여러 개를 넣어 복잡성을 분산, 효율up
2. 캐시 level이 존재

Many Integrated Core(MIC): core가 10~20개 짱 많은 경우

Graphics Processing Unit(GPU): 그래픽 처리용

1. Vector 프로세스를 통해 복잡하고 반복적인 계산을 많이 한다

Amdal’s Law(암달의 법칙)

1. 프로세스(CPU) 하나 쓸 때와 여러 개 쓸때의 실행시간 비교
2. 소프트웨어가 프로그램에 각 프로세서를 잘 배분해야 시간이 빨라진다. 따라서 소프트웨어의 병렬 수행 적용이 중요하다는 것
3. Speedup = Original/Advanced = T/((1-f)T+fT/N)

* T: 1개의 프로그램 실행 시 걸리는 총시간
* f: CPU가 쓰는 시간 (비율로 나타낸다. 1-f는 cpu랑 상관없는 시간(메모리 시간 같은..))
* N: 코어의 수

1. f값에 영향 많이 받음(CPU, MM 조화 중요)

Clock에 대한 이해 완벽 정리

CPU의 1초당 클럭 사이클 수: **Clock Frequency, Clock Rate(클럭 프리퀀시, 클럭 속도)**

* 1초당 cycle이 몇 번 반복되는지 ex) 2GHz(1초에 2x10^9번 반복됨)

Clock cycle 1개당 몇 초가 걸리는지: **Clock Period, Clock Cycle Time(클럭주기)**

* 1/clock frequency

처리 시간(T) = Ic x CPI x τ = Ic x (p+m x k) x τ

* Ic: 수행된 명령어의 개수
* CPI: 하나의 명령어 수행 시 필요한 cycle 개수
* τ : 단위
* CPI x τ : 평균적으로 하나의 명령어 수행 시 걸리는 시간
* P: CPU에서 처리하는데 걸리는 시간
* m: 메모리 접근 횟수
* k: MM에 한 번 접근할 때 걸리는 시간

\*CICS, RICS 모두 이 처리시간을 줄이려고 노력한다. CISC는 명령어 개수인 Ic를, RISC는 명령어 자체는 복잡하지 않아 대신 처리할 명령어가 많아지고 따라서 명령어를 처리하는 시간을 줄인다.

좋은 CPU일수록 처리시간이 짧다. 따라서 컴퓨터의 성능 측정을 위해 이것저것 다 때려 넣은 benchmark program 이라는 프로그램을 여러 번 수행하여 처리시간의 평균치를 구한다.

이때 평균치는 산술평균과 기하평균을 통해 구한다.

* 산술평균: 실제 측정한 시간 평균치 구할 때 good

실행시간 모두 더함 / 횟수로 나눔 ex) (a+b)/2

* 기하평균: 상대적인 실행시간 평균치 구할 때 good

실행시간 모두 곱한거에 루트 씌움

처리율(throuput = MIPS): 1초당 처리하는 정도를 million 단위로 나타낸 것

= 수행된 명령어 개수/s

=(기하평균)^2 / 산술평균

benchmark program 의 원칙

1. High-level 언어로 쓰여져야 함
2. 어떤 컴퓨터에서든 수행 가능해야 함
3. 여러 일 수행 가능해야 함
4. 측정 쉬워야 함
5. 쉽게 다룰 수 있어야 함

Ex) SPEC(system performance evaluation corporation): 인위적 benchmark program 이므로 이를 기준으로 CPU 성능을 높이다 보면 실제론 적용되지 않을 수도 있다

Chap3. 컴퓨터의 기능과 상호작용을 개괄적으로 바라보기

* 현대의 컴퓨터는 폰 노만 구조 가 기본 베이스임(IAS)
* 프로그램 수행 전 데이터와 명령어를 메모리에 저장하고 사용
* 주소를 가지고 데이터와 명령어를 참조
* 순차적으로 실행
* Hardwired program (하드웨어로 된 프로그램) -> 이건 RISC의 특징 아닌가….

Programming in hardware : 정해진 연산을 정해진 순서대로 하도록 하드웨어에 프로그래밍

Programming in software : 명령어 해석기와 실행장치가 소프트웨어 있는 프로그램을 수행

소프트웨어

1. 명령어/코드들의 연속
2. 명령어 해석
3. 제어신호 생성

하드웨어

1. CPU: 명령어 해석기, 범용 산술 연산 기능

다양한 register 존재 1) MAR(다음에 처리할 특정 메모리의 주소를 저장)

2) MBR(메모리로부터 읽은 혹은 메모리에 쓸 데이터 담고 있음)

3) I/OAR(I/O 장치 주소)

4) I/OBR(I/O모듈 데이터) => 느린 I/O장치의 시간적인 지연 조절

1. I/O 구성요소: input/output 모듈
2. 메모리

Instruction Cycle(하나의 명령어를 수행하는 과정)

1. 명령어 주소 계산
2. 명령어 fetch
3. 명령어 디코딩
4. 명령어 수행 시 필요한 데이터 주소 계산
5. 명령어 수행 시 필요한 데이터 fetch
6. 데이터 처리(operantion), 수행
7. 결과를 저장할 주소 계산(메모리에 저장될 수도 있고, 다음 명령어에 쓰일 수도 있음)
8. 저장

Fetch cycle : 명령어를 가져오는 과정

1. 메모리로부터 현재 수행할 명령어가 있는 메모리 주소 가져와 MAR에 저장하고 다음에 수행할 명령어의 메모리 주소 PC에 저장
2. M[MAR] 해서 실제 명령어(데이터) MBR에 저장
3. Fetch된 명령어는 IR로 들어가 load 되고, 이때 PC가 증가하며 반복

명령어 실행 시 수행되는 것

1. 프로세서와 메모리(I/O 모듈)간의 데이터 변환
2. 프로그램 실행 순서등을 바꿔주는 제어 발생(분기)
3. 데이터 프로세싱(산술, 연산)

Interrupt

1. 프로그램 수행 중 의도치 않게 해당 수행 멈추고 다른 수행해야하는 경우 발생
2. 프로그램에서 수행 불가, 시간 지연, 하드웨어적 오류, 느린 I/O장치 위해 발생
3. 어떤 경우에 어떤 interrupt가 발생해야 하는지 다 프로그래밍 해둠(interrupt Handler가 제어)

Ex) I/O장치 준비되는 동안 다른일 수행하고 준비되면 interrupt 걸어서 수행

1. Instruction cycle 후에 interrupt 체크
2. Interrupt 안에 interrupt 발생할 수 있음

I/O 함수

1. 프로세서와 I/O 모듈이 데이터 읽고 쓰는 내용 주고 받음
2. 각 장치들은 주소로 구별
3. 프로세서가 I/O모듈에게 권한주어 CPU는관여 안 하고 모듈이 직접 메모리와 교류

(direct memory access