2024.07.03. 회의 필기.
MPU를 위해서는 multiply add 넣어야 하지 않을까? 엣지 브라우징 관심 있다...
유선 인터럽트 보단 privilege 더 어렵.
cache쓴다 했는데 더 크게 쓰진 않는다?
파이프라인 슈퍼 스칼라 구현 하면 될거 같은데?
우리가 구현하는 게 버스 같은 것이다.
ARX?AIX 같은걸 구현하는건 난도가 어려워서 한세대 전 것을 구현할 것이다.
메모리 자체를 버스로 구현하면 오케이,
MMU는 메모리 접근과 한사이클? 다른 사이클? > 무조건 한 사이클에 끝나야 한다.
인터럽트 구현이 배우는 내용과의 차이로 어려운 점이 있다.
ISA 확장만 참고만 해서 구현한다?
인터럽트가 발생했을 때 작동 중지후 작업 해야하므로 권한 높아야 함. > OS가 제어권 가지고 유저가 OS 제어권 가지려는거 막는 게 난이도가 어려워진다.
하던일 멈추고 다시 실행하거나, 넘어가거나 하는 행위는 가능한데, 종류가 많다.
인터럽트에 필요한 벡터용 레지스터 또 만들어야 함. 벡터를 위한 숫자를 위한 레지스터도 필요함. 쉬운 거 먼저 하고 나중에 소소한거 한 두 개만.
뭐가 필요하다고 생각하는가?> 곱셈 나누셈, 부동소숮ㅁ을 위한 csr확장자(꼭 부동소수점 아니어도 필요함.), 명세에 crypto 확장 존재. > 꼭 필요하지 않음. 곱셈도 확장이지 반드시 필요하지 않음.
우리가 공통적으로 cpu에 올리려고 하는 것이 구체적인 합의가 무엇일까?
OS는 안올림 OS가 올라갈 수 있게는 만들되, 올리는 프로그램은 베어메탈로 할 생각.

서로 생각하는 최소 요구 cpu 기능.
risc-V는 프리빌리지와 유저로 나뉘어 명세가 설명되어 있음.
톡방에 올라간 RISC-V 자료 참조:
RV32I는 우리가 포함해야함. 최소
E,64I,128I는 포함 안해도 될듯.
chapter 6,7은 꼭 하고,
11,12제외. 뭐 없이 구현하는 법.
13은 꼭 해야함.
14a-아토믹 인스트럭션 없으면 원래 되면 안되는 일이 생김. 한 프로세스를 처음부터 끝까지 한 프로세서가 처리하게 하는대 필요. 꼭 있으면 좋게쌈.
15,16 제외
17 멀티 코어용이라 제외.
19 CMO, 캐시 메모리 ... 굳이 필요없다.
20~25 floating point. f만 하면 좋겠다. 어차피 버스 32비트. 굳이 64비트 해야할 필요성 못 느껴서 d, 등 필요없을듯.
이후 내용들은 영 필요 없는거 같아서 제낄 것이다..

M,A,F는 이번회의에서 확정짓지는 않을 것이다.
I에서 대부분 배운 것이 많을 것이다. R타입, NOP인스트럭션, 등.

27??,28??인스트럭션,29힌트 인스트럭션 존재. 메모리 오더링 경우 race condition 발생 방지 위해 필요. 확률적으로 될지 안될지 알 수 없다.
우리는 싱글 코어만 구현하는데 굳이 필요할까.
2.7 맨 앞 other 인스트럭션
2.7은 제외를 하자.
2.8 ZrCSR?..? 좀 볼 필요 존재.
힌트는 명령어가 아니다. 캐시한테 힌트 주는 명령어로 보면 됨. 기존에 있는 add,end 이용.
2.6은 반드시 존재.
prediction은 반드시 필요.
privilege의 경우 risc_V 그대로 따라가기엔 역부족에 다들 동의.

그러니 어떻게 구현할지 리스팅을 해보자.
박제윤 선배님도 명확하게 정해지지 않음.
> 권한을 다르게 취급해야하는게 뭐가 있는가?
I/O 인스트럭션, (trap과 인터럽트) 위한 연결 각각 필요.
..? 필요 CSI 종류 다양함. RISC-V는 최대 4796?4976? 개 쓸 수 있게 해 놨음.
cpu에러 발생시 핸들링.

코드짤 때 명령어 잘못짜서 없는 명령어 실행하는 경우 에러.
exceptional flow 3가지 있었음.
exception 종류 두 - 가지 있었음. aschncronus(trap, falut, ...)...interupt 구현 필요.
context switch,
..?

실제로 우리가 권한이 필요한 부분 > I/O. 메모리 쓰기는 유저든 관리자든 상관 없음.
트랩 호출> 시스템 콜 순간 프리빌리지로 넘어감. I/O만 신경쓰면 될거 같다.
인터럽트를 제외하곤
우리가 하는거 디스크에 쓰자!

요약하면 우리 목표
트랩, 인터럽트, 폴트, 이 세 가지에 대해 메카니즘 구현 가능한 레지스터 만들어야 함.
버추얼 주소 위한 레지스터 구현 필요. > VA에 구현 구조 중 난도 높은거: 세그먼트 테이블..? 테이블의 주소 알려면 테이블 주소 필요 > 해당 주소 저장하는 레지스터도 필요.
NCR, X86 그런거 있더라.
컨트롤 레지스터 CR2,CR3 같은 거
CR4,CR8은 필요 없다. 디버그 레지스터 필요 없다.
유선 인스트럭션에 필요한건 이 정도인거 같다.

일단 지금까지 우리가 논한거 CPU 외부. CPU 내부는 논하지 않음.
이렇게 애매해지는 이유는 이제야 구현 기능 논하기 때문.
위에 언급한 특수레지스터들과 디폴트 레지스터, 리턴 레지스터, 32개짜리 두 개, 등.

CSR 비스무리 한 거 보고 참고하자. > 컴구 외적인 부분.
실제 관심을 가지고 있는 부분들 같은 부분: 실제 분기에서 정책은?, 파이프라인...일텐데, 그런 부분은 다음 회의로.

그러니 지금은 I타입에서 논해보는게 맞는듯.
이준호 선배님 노트북 안 정리하신 명령어 파일 보고 포함할지 말지 고민하는 중.
무슨 쉬프트는 있어야 한다..
if문 구현하는 명령어들에서(이것만 있으면 등호만 판단한다는 이야기.).. 너무 넘치면 언사인드 빼겠다.

..?

펜스 확실히 필요 없음
이퀄리 필요하니 놔 두고.
111111111하고 가냐 000000000하고 가냐에서 차이가 난다. 비트 이동으로 곱셈, 나눗셈 하는 이진법상..

...

루트연산은 일단 빼자. 계산 너무 어렵.
dsp에선 ..?
FMI..? 하여튼 합성곱 빼자.
변환도 제거.

일단 있는것도 해보고 안올라가면 줄여야함.

다음 회의까지 할 일:
1. CSR, 아토믹 제대로 아는 사람이 단 한명도 없다. >
하기로 한거: M, I,
펜스, csr, A(아토믹)은 고학년 세 분이서 나눠서 조사한다.
이준호 선배님 펜스
서승현 선배님 csr
박제윤 선배님 아토믹

? 스칼라?
캐시 정책
L2 안쓴다고 한것: 압도적인 큰 L1이 있으면 좋다. > 다만 이러면 원하는 캐시 블럭 찾는데 시간 걸림. (다만 현실적으로는 L1소자가 L2 보다 비쌈.)
TLB는 이미 답이 나와있다. 어떤 시스템에서건 fully associated. TLB는 캐시하면서 같이 하면 될듯. > 조사를 하고 생각을 해보면 좋을 것 같다. 5단계가 국룰이지만 다른 의견이 있을 수 있으니까. 특히 부동소수점 경우 시간이 많이 걸려서 스테이지를 더 나눌 수도 있다. 여러

사이클을 쓴다던가. 캐시 한라인씩만 들어가는거 > 다이렉트 매핑을 쓰는게 좋은가? 아니면 어소시에이트를 쓰는게 좋은 지 조사 필요. 어소시에이트가 커질 수록 제어 로직이 커진다.
박제윤 선배님.모토로라 MCU 이런게 있다. 관련 서적 읽고 정리한게 있어서 다음 회의까지 정리해서 올려 주겠다.

다음 회의: 7.18일 목요일 18시 이후. 저녁 먹고 올 사람은 저녁먹고 와도 오케이.
이후 회의 내용을 문서화를 하자. 그렇게 리스트 만들고 작업에 착수하면 될듯.

저학년 팀: 베릴로그 공부. 베릴로그 안되면 아무것도 못하니 일단 깔아서 확인해봐라.
일단 부족한 공부 따라가보기.
결론:
다음 회의: 7.18일 목요일 18시 이후. 저녁 먹고 올 사람은 저녁먹고 와도 오케이.
이후 회의 내용을 문서화를 하자. 그렇게 리스트 만들고 작업에 착수하면 될듯.

다음 회의까지 할 일:
1. CSR, 아토믹 제대로 아는 사람이 단 한명도 없다. >
확실하게 하기로 한거: M, I,
펜스, csr, A(아토믹)은 고학년 세 분이서 나눠서 조사한다.
이준호 선배님 펜스
서승현 선배님 csr
박제윤 선배님 아토믹

저학년 팀: 베릴로그 공부. 베릴로그 안되면 아무것도 못하니 일단 깔아서 확인해봐라.
일단 부족한 공부 따라가보기.

추가 사안:
박제윤 선배님: 모토로라 MCU 이런게 있다. 관련 서적 읽고 정리한게 있어서 다음 회의까지 정리해서 올려 주겠다.