2024.07.18. 19시 회의.
지난 회의 결론 및 할일 정리:
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 정리한 게 있어서 다음 회의까지 정리해서 올려 주겠다.

회의 시작:

저학년 조: 해야될 명령어가 정해지면, 그것에 대응하는 기계어를 정해주는 어셈블러를 제작하기. 파이썬을 통해 문자열을 잘라서 만드는 형태라던가. 일종의 어셈블리어에 대한 컴파일러인 셈. > 그게 어셈블러다.
ex: add > 011011000 등..
더블 스페이스 구분 힘들면 스페이스 한번으로 제한하는 걸로 제한 조건 걸어도 된다. 어떻게든 만들어와라.
질문: txt파일로 만들어지길 요구하는 것인가?

조사내용 발표 - 자료 참조하기.

펜스 조사결과 필요없는 걸로 판명. > 구현 안하는 걸로.

CSR: 시스템에 있어서 상태 저장하는 레지스터
메모리 저장 형태: 주솟값, 소스, 명령어(function), 목표, CSR_offcode.
사용자 레지스터 맨 앞 비트 4개로 결정...?
트랩 발생시 트랩 벡터 저장하는 거. 등등 있다..?
mstatus 같은 건 당연히 필요하지만, band 뭐시기 같은 제조사 나타내는 레지스터는 굳이 필요 없다.
파편화 때문에 필요한 것들, 필요없는 것들이 생긴다.

레지스터도 마찬가지. 따라서 이번 회의를 통해 필요한 레지스터나 명령어, 필요없는 것을 가려낼 것이다.
파이프라인 같은 거 예외처리 이야기에서..
명령어가 단일하면 아토믹이라 생각하면 될 듯하다.
예외처리하다 일이 멈추고 관련 데이터 변경되면 단일 명령어면 괜찮겠지만, 여러 명령어면 오류가 생길 것이다.
funct3 4번 항목은 굳이 구현 안해도 레지스터 이용하면 구현 안 할수 있다.
일단 1,2,3번 항목만 구현해보고 생각하는 걸로.

아토믹:
lr.w, sc.w가 있는데, 멀티 스레드에서 싱크로 나이즈 위해 있는 기법.
결론: 얘네는 필요하다. reservation 적용에 필요.
아토믹 스왑, atomicswap: 스왑하는 거. 앤드나 오어 같은거 아토믹하게 수행해주겠다.
명령어 수행할 때 이게 아토믹한지 아닌지만 보면 될 거 같다.
다 끝내고 수행하면 되니까?
and,or등등 이 외의 것들은 아마 안하게 될지도..
load reserved, sc.w, swap는 반드시 구현.

이제 어떤 구조를 만들지에 대한 이야기를 해보려 함.
슈퍼스칼라 파이프라인은 몇 개이며, 각 단계는 어떻할지, 캐시는 몇 개 넣을지, 레지스터는 몇개할지 등등.

지금 논의 하는게 큰 의미가 없을 수도 있다. 원래 이건 성능을 위해 고민하는 건데, 우린 아예 처음이라 성능 개선을 만드는 게 없다.
정수 파이프라인:부동 소수점 파이프라인 비율에 따라 몇 개 넣을지, 그걸 정해도 FPGA에서 돌아가는 지도 중요하다. 조사한거나 참조한거 있다면 그거 따라 만들어 보는 걸로
일단 박제윤 선배님 조사자료 참조하기.
...
ideal computation 고려 필요.
load 파이프라인과 ideal 파이프라인 개별적으로 만들 필요성.
fetch decode 오퍼레이션 가져와서 ....?
5,6단계
어드레싱 모드가 심플해야한다. 애드할 때 메모리 접근할 수 있다. 그러면 복잡해지니 디코딩을 쉽게하기 위해 필요.
..?> 리스크 파이브라 고려 필요 없음.
파이프라인을 깊게 팠을 경우.
fetch에 따른 멀티포트 메모리 필요. > 자원이나 비쌈 등등.. 블락드 램은 싱글포트일 가능성.

캐시 정할 때 스플릿 캐시냐 유닛 5 캐시냐 나뉨.
l1 캐시 5조각으로 나뉨?
이하 파일 내용 설명이라 더 자세한 이야기는 pass.

명령어 타입은 무엇으로 나누는가..

포워딩 패스는 구현하였는가? 했으므로 패스.

슈퍼스칼라> 아예 안해봄. 이야기를 진행하면.. 펜티엄 5의 경우 파이프라인이 깊다.
...
각 파이프라인이 특화된 스킬을 다루게 된다?

아웃오브오더를 하되 4개나 5개로 줄여보는 것도 나쁘지 않을 것 같다.

지금 고려하지 않을 것들에 대한 이야기:
...?
캐시라인이 맨 끝에 2개가 있는데 하나는 메모리에 있는 경우에 대한 이야기. > 대역폭 효율성 등등.. > 얼라인먼트 하드웨어가 필요해짐.
디코드에서 자원 많이 필요해짐.
트위디 코딩? 안하는 추세. 이런 기법이 있다. 메모리에서 i캐쉬로 로딩할 때 부분적으로 디코딩하고, fetch했는데 브랜치면 바로 넘긴다..?
디스패치? 싱글파이프라인이 여러개인 경우..
디코드는 됐는데, 연산자가 저장이 안된 경우. > 버퍼 필요.
뒤에것이 의존성이 없으면 아웃오브오더 이용, 또다시 버퍼 필요.
레저베이션 스테이트라 한다?
디스트리뷰티드..?
..?
옆에 정수 파이프라인

실제 익스큐션시 고려해야할 사안.
어떤 정수 파이프라인 2개가 존재 하나만 amd 까지 처리. 부동소수점 파이프라인 하나.
로드용, 브랜치용 하나씩 만드는것 보다 나을지도?
두개 이상의 멀티 포트로..?

자원 문제, 복잡도 문제도 있어서 파이프라인은 3개 이상으로 늘리지 않았으면 좋겠다.

??는 아직 고려하지 않아도 되고.

exception의 예시.

..?

인스트럭션 자체가 시스템 콜인 경우

결론:

개략적으로, 슈퍼스칼라 파이프라인 프로세서에 대한 설명이었음.
앞으로 고민해야할 질문: 파이프라인 몇단계? 몇 웨이? 그 안에서 익스큐션 파이프라인은?
단, 익스큐션 파이프라인은 사람마다 명령의 구현방식이 다르기에, 명령어 구현하며 정해도 될거 같다.
파이프라인의 깊이는 우리가 싱글 파이프라인 프로세서를 만든다고 고려하고 고민하자.
어차피 우리가 고민해야하는건 익스큐션 파이프라인이다.
페치와 디코드를 합친다던가. 등등. 페치가 기능을 막 붙이면 쉽지가 않다. 가상접근은 메모리 주소 접근할 때마다 이루어진다. 보통은 캐시도 물리주소를 쓰는 편이다. 가상주소를 써버리면 프로세스는 둘다 0번을 쓰는데, 캐시가 0번만 보고 다른 프로세스 껄 가져다 쓸 수있기 때문. 이러지 않으려면 추가 정보 입력 필요.
또한 이러면 fetch할 때마다 인스트럭션 ?? 접근 필요.

박제윤 선배님: 기본으로 6스테이지로 생각하는 건 어떤가? 메모리 뭐 빼면 5스테이지.

일단 깊이 파이프라인 파자는 의견은 없는걸로.
캐쉬는 i,d로 쪼갤것인가 아님 하나로 할 것인가? > 일단 캐쉬는 i캐쉬, d캐쉬 쪼개는걸로.

몇 웨이로 하는가? 부동소수점은 무조건 하나 필요. 복합정수와 단순정수? 아님 로드스토어와 다른거? 하나 할 것인가? 성능이 높을 것으로 예상되는 거: 복합&단순.
구현의 어려움을 고려하면 둘다 비슷할 것으로 사료됨.
>복합 & 단순으로 일단 해보는 걸로.

지금 안정해도 될것같다 싶은것:
아웃오브오더의 디그리 > 몇 개까지 명령어 넣을 것인가? 하려면 3~4개 가능.
부동소수점 파이프라인 깊이 > 부동소수점 실행의 인스트럭션이 나와야 함. 맨 앞에 있는 것에 대한 의존성은 계산 과정이므로 시간이 소요. 깊어지면 당연히 시간 오래 걸림. 따라서 계산 방식=인스트럭션이 정해져야 깊이를 얼마로 할지 정해질 것으로 생각됨.

디스트리뷰티드와 센터럴라이즈 아베스? 하나의 파이프라인 모니터링하는 것과 여러 파이프라인 모니터링 하는건 다를 것 같다. 더 쉬운건 디스트리뷰티드일 것 같다.
3웨이면 디스트리뷰티드가 편할듯.
일단 디스트리뷰티드로 잡아두고 해보는 걸로.
유명한 프로세서들은 센터럴라이즈 쓴다는데(연산이나 자원측면에서 유리.), 문제는 참고하는 책이 펜티엄 4시절 책.

캐쉬 구조랑
다이렉트로 하는데 문제가 있는 사람? 풀리 어소시에이티드로 하면 로직이나 할 일이 더 복잡해진다. > 다이렉트로 하는 걸로.

...? > 쓸 수 있는 비렘 양 보고 고민해보자.
이건 소프트웨어의 로켈리티에 따라 갈린다. 우리가 쓸수 있는 비렘 캐쉬 이런게 작아버리면

금방 내려갈 수 있다... 자료 좀 더 조사해보고 정하는 걸로.

결론:
다음번까지 해야하는 거:
명령어 셋 다 정함. > 누군가 사용할 명령어 리스트 만들어야함.
레지스터 목록도 정리 필요. R15까지 다 넣고, CSR이 좀 문제가 됨. CSR 부분이 복잡해서 일거리.
결론적으로 컴구 책 마냥 그림이 그려져야 함. 그림을 한 사람에게 맡기긴 너무 큰 것 같다. 각각의 컴포넌트에 맞춰서 그려야 할 것 같다.> 다음 할 일

분게이츠 알고리즘 뭐 쓸건지.,
비렘 어떻게 쓸건지 알아봐야함.
메모리랑, 로드스터 유닛 뭐로 할건지 - 버스 배운걸로 할건데 그 구현이 상당히 복잡. 프로토콜 조사 필요.
슈퍼스칼라 내용: 토마큘러스 알고리즘 등 매우 많다. 어드밴스드 알고리즘 뭐 있는지 책에서 찾아볼 필요 있음.
브랜치 프레딕션, 버스 쓰는 법, ...조사

박제윤 선배님: 버스 사용법, 슈퍼스칼라의 어드밴스드 테크닉스.
이준호 선배님: isa 하는 사람 어드레싱 모드 뭐하는지까지 해야함(= 명령어 정리.) 비렘관련 조사.
서승현 선배님: 레지스터, 분기 알고리즘.

티엘브이> 엔트리 개수 관계 없음. 그때 가서 생각하는 걸로.

저학년 어셈블러는 기한은 없다. 편의성 용.
텍스트>텍스트면 오케이.

실질적으로 가상주소를 쓰는 게 테스트 프로그램이 해봐야 할거는 페이지 프레임 만들어서 가상주소 잘 돌아가는지 해보고 등등. 가상주소 자체는 구현할줄 몰라도 만드는게 쉬워서 이론만 이해해도 괜찮을것 같다.
하드웨어 접근 같은 것은 지난 작품에서 떼오면 될 것으로 생각됨.
글씨 띄울 생각이 막막함. > 미리 저장해놓고 오프셋에 맞춰 위치 띄우기만 하면 된다...
매번 입력들어오면 인터럽트 받아서 띄워야한다.

리스크 5는 ??? 이용하죠. i 주소쓰는데 유저모드이면 오류 발생. 유저 프로세스가 실행중이

면 cpu에 표현됨. OS로 권한이 넘어간 순간 CPU 비트가 바뀌어서 프리빌리지드인걸로 알려줌.

다음 번 미팅: 다음 할 내용에 따라 그냥 그 시간까지 톡방에 올리는 것도 나쁘지 않아보임. 다음주 화요일 ~ 목요일까지 데드라인. 파일 업로드 이후 이틀 동안 올라온 정보 숙지 후 회의 개최.

다음장: 회의록 총 정리.

바라미 FPGA - CPU 구현 작품 팀 2차 회의

Ⅰ 회의 일시:

2024.07.18.(목) 19:00~20:30

Ⅱ 회의 참석자:

박제윤, 이준호, 서승현, 김희준(회의록 작성자).

Ⅲ 지난 회의 요약:

배정되었던 할 일:

RISC-V 확장 조사:

이준호 선배님: Fence

서승현 선배님: CSR

박제윤 선배님: Atomic

저학년 팀:

베릴로그 공부. 베릴로그 안되면 아무것도 못하니 일단 깔아서 확인해봐라.

일단 부족한 공부 따라가보기.

추가 업무:

박제윤 선배님: 모토로라 MCU 정리 및 공유 (보너스.)

Ⅳ 회의 내용:

배정 받았던 일의 결과 공유:

Fence: 이준호 선배님 - 작품제작에 큰 의미가 없을 것으로 판단. 작품에서 제외.

CSR: 서승현 선배님 - 카톡방에 조사내용 정리된 CSR.pdf 업로드 됨. 참조 권장.

funct3 항목 중 CSRRW, CSRRS, CSRRC 세 가지만 우선적으로 구현.

나머지(CSRRWI, CSRRSI, CSRRCI)는 앞의 세 funct3의 즉시 값(5비트 사용)이나 레지스터를 활용하면 대체 가능하여 구현 안해도 괜찮다고 판단.

> 따라서 사용할 ISA 작성 시 CSR의 경우 필요 레지스터의 신중한 고려 필요.

조사 내용 요약:

CSR: 시스템에 있어서 상태 저장하는 레지스터

CSR 주소 구조: 주솟값, 소스, 명령어(function), 목표, CSR_offcode.

Atomic: 박제윤 선배님 - 카톡방에 RV32A instructions.docx 파일 업로드. 참조 권장.

Multihead synchronization을 위해서 Load reserved필요.

> 따라서 Load reserved를 위해 아토믹 중 lr.w, sc.w는 구현 필요.

이 외에 amoswap나 amo+OP명령어들의 경우 여유가 되면 하고 당장은 하지 않음.

조사 결과 결론:

조사 내용 바탕으로 CSR과 Atomic 확장에 대한 ISA, Register 목록 작성 필요.

어떤 구조를 만들지에 대한 개략적인 토의:

파이프라인을 몇 단계로 구성하는가?
본래 성능 개선을 위해 고려하나 우린 아예 처음 만드는 중. > 일단 국룰 5~6단계.
fetch > decode > oper ram 가져와서 연산, store + load/store위한 메모리 접근 stage.
단계가 너무 길어질 경우 fetch가 2단계 이상이 되어 멀티포트 요구 가능성 존재 -

슈퍼스칼라 파이프라인은 몇 개로 구현하는가? > 복합정수, 단순정수, 부동소수점 하나씩.
비율을 정해도 FPGA의 성능이 받쳐주는지 고려 필요.
로드용 파이프라인과 애드용 파이프라인을 따로 만들지에 대한 고민 필요. > 자원적 한계.
인디펜던트 컴퓨테이션스 - 의존성 줄이는 거

Out of order의 degree: 명령어 3~4개 정도 예상.

Centeralized와 Distributed 중 Distributed로 채용.

캐쉬 구조: I캐쉬, D 캐쉬 쪼개어 사용.
주소 이용은 Direct(캐쉬에서도 물리주소 직접 이용)로 사용.

ISA 작성과는 별개의 이야기: Addressing mode의 단순화 필요.

메모리랑, 로드스터 유닛의 구현.
> 지난번에 올린 세미나의 버스로 구현하려함. 구현이 상당히 복잡. 프로토콜 조사 필요.

가상주소 자체는 구현할 줄 몰라도 만드는 게 쉬워서 이론만 이해해도 무난.

당장은 아니지만 앞으로 고려해야 하는 일들:
티엘브이> 엔트리 개수 관계 없음. 그때 가서 생각하는 걸로.
결론적으로 컴퓨터 구조론 책 마냥 큰 그림이 그려져야 함.
그림을 한 사람이 다하기엔 부담. > 각자가 각자의 부분에 맞춰서 그려야 할 것 같다.
Branch Prediction
- 하드웨어 접근 > 지난 작품에서 차용하면 될 것으로 생각됨.
글씨 띄우기 > 미리 저장해놓고 오프셋에 맞춰 위치 띄움.
매번 입력 들어오면 인터럽트 받아서 띄워야함.
- 가상주소 테스트 프로그램이 할일: 페이지 프레임? 만들어서 잘 돌아가는지 확인.

이 외의 내용은 이해가 부족하여 정리하지 못하였습니다..

Ⅴ 회의 결론:

작업 마감: 7/25(목). 작업 내용은 카톡방에 올리기.

다음 회의: 전원 작업 업로드 후 이틀 뒤 회의. 업로드 된 자료 숙지 요구됨.

작업 분배:

박제윤 선배님:

- 메모리-로드스터 연결에 대한 Bus 구현 프로토콜 조사,
- SuperScalar의 Advanced Technics, 관련 구조들 조사(토마큘러스 알고리즘 등.).

이준호 선배님:

- ISA(구현할 명령어 목록) 정리하여 작성,
- Implantation관련 B ram 사용법 조사.

* ISA 작성 시 Addressing mode 뭐하는 지까지 정리 필요.

서승현 선배님:

- Register 목록 정리, 분기 알고리즘 조사.

Register 목록 작성 시 CSR 확장에 대한 세심한 고려 요구됨.

저학년 팀:

- Test Bench 제작 가능할 정도로 Vivado 및 verilog 학습 필요.
- 편의성 증진을 위한 어셈블러 개발. 기한 없음.

차후 정리된 ISA 목록을 바탕으로 txt > txt 형태의 명령어 > 기계어 변환 코드 짜보기.

+ 문법 검사 기능 필요.

ex: add > 0100101001

앞으로 고민, 배정해야 할 일:

- TLV > 엔트리 개수 관계 없음. 그때 가서 생각하는 걸로.
- 결론적으로 컴퓨터 구조론 책 마냥 큰 그림이 그려져야 함.

그림을 한 사람이 다하기엔 부담. > 각자가 각자의 부분에 맞춰서 그려야 할 것 같다.

- Branch Prediction
- 하드웨어 접근 > 지난 작품에서 차용하면 될 것으로 생각됨.

글씨 띄우기 > 미리 저장해놓고 오프셋에 맞춰 위치 띄움.

매번 입력 들어오면 인터럽트 받아서 띄워야함.

- 가상주소 테스트 프로그램이 할일 - 페이지 프레임? 만들어서 잘 돌아가는지 확인.