[프로그램 동작]

(단, 수행 시간이 길기 때문에 동작 확인의 편의를 위해 하이라이트 참조하여 마지막만 실행)

→ 중간 과정에서 생성되는 파일들(대부분 트레이스/데이터셋)의 경우, 대용량이므로 첨부X

1) 가상 & 물리주소 변환
   A. simulator/traces/extract_trace.ipynb 실행하여 .vaddr 추출
   B. simulator/traces /v2p.py 실행하여 .paddr 추출

2) 시뮬레이터 실행하여 miss에 대한 .cstate 추출
   A. simulator/settings.py 에서 옵션을 사용 트레이스에 맞게 변경
   B. 학습을 위해 세가지 버전을 추출
      i. 5,000,000개의 Demand에 대한 매 시점에서의 .cstate
         → vocab 구성용
         이 경우, cache.py 내 access 함수 수정 필요
         → hit, pf_hit, pf_miss인 경우 모두에 대해 cstate 저장하는 코드 주석 해제

      ii. 4,000,000개의 Demand에 대한 demand miss 시점에서의 .cstate
         → valid 데이터셋 구성용
         이 경우, cache.py 내 access 함수 수정 필요
         → pf_miss인 경우에 대해 cstate 저장하는 코드 주석 해제

      iii. 3,000,000개의 Demand에 대한 demand miss 시점에서의 .cstate
         → train 데이터셋 구성용
         이 경우, cache.py 내 access 함수 수정 필요
         → pf_miss인 경우에 대해 cstate 저장하는 코드 주석 해제

   C. simulator에서 python3 main.py 3 즉, 옵션3 선택하여 Leap 실행
      i. simulator/deep_learning_data/에 .cstate 파일 생성됨

3) 모델 학습
   A. clstm/clstm_main.ipynb 파일에서 Setting environment variable 설정 후, 학습 진행

4) Test 트레이스 구성
   A. 5,000,000개의 paddr 중 뒤 1,000,000개 뽑아서 사용
   → 동작 테스트를 위해 100,000개 데이터 가진 bc_test_64.paddr 만들어 둠

5) 모델 추론
   A. clstm/clstm_main.ipynb 파일에서 Test 부분이 추론 과정임
   B. 아래 Idx to Addr까지 완료하면 Test 트레이스에 대해 추론한 결과 나옴 (.csv)
   → 동작 테스트를 위해 clstm/results/bc_test_result_addr.csv 만들어 둠

6) CLSTM 프리페치 결과 확인

A. simulator/main.py 에서 clstm 결과 파일 주소 설정

B. 시뮬레이터에서 python3 main.py 5 즉, 옵션5 선택하여 CLSTM 실행
단, CLSTM90 사용 시, cache.py 내 access_clstm 함수에서 clstm90에 해당하는 코드 주석 해제 & 바로 아래 clstm에 해당하는 코드 주석 처리

C. CLSTM 모델 추론 결과를 읽어와 시뮬레이터 동작

D. 결과 확인

➔ 편리한 동작 테스트를 위해 위 세팅을 모두 진행해 둔 상태

➔ 즉, simulator/ 에서 python3 main.py 5 실행 시, clstm의 실행 결과 볼 수 있음