제 1 교시

국어 영역

성명		수험 번호									

[1~3] 다음 글을 읽고 물음에 답하시오.

중앙 처리 장치(CPU)는 컴퓨터 시스템을 통제하며 프로그램을 실행하고 처리하는 가장 핵심적인 제어 장치를 말한다. CPU는 명령어를 사용하여 컴퓨터 하드웨어를 통제하고 작동시키는데, 이 과정의 평균적인 빠르기는 CPU마다 제각각이다. CPU와 같은 디지털 회로가 작동하기 위해서는 일정한 간격으로 전기적 신호를 공급받아야 하는데, 이 신호를 클럭이라고 부른다. 이 클럭의 시간 간격을 클럭 사이클이라고 하며, 1 클럭 사이클당 소요 시간인 클럭 주기에 명령어 1개를 실행하기 위한 평균 클럭 사이클 수인 CPI를 곱한 값이 작을수록 CPU의 처리 속도가 빠르다고 할 수 있다.

가장 기초적인 CPU는 단일 사이클 방식을 사용한다. 단일 사이클 방식 CPU는 1 클럭 사이클마다 하나의 명령어를 수행하도록 설계되어 있다. 명령어는 인출, 해석, 실행, 기억 장치 접근, 그리고 쓰기의 단계를 거쳐 실행된다. 인출은 다음 실행해야 할 명령어를 외부 기억 장치로부터 CPU에 적재하는 단계이다. 항상 다음 실행할 명령어가 저장되어 있는 기억 장치의 주소 값을 가리키도록 갱신되는 프로그램 계수기(PC)를 통해 이 단계를 수행할 수 있다. 해석 단계는 이전 단계에서 인출한 명령어를 사전에 약속된 규칙에 따라 해독하여 어떤 값을 가지고 어떤 동작을 수행해야 하는지 결정하는 단계이다. 가령, ⓐ CPU 내 기억 장치 1번지의 값과 2번지의 값을 더하여 3번지에 저장하라는 명령이 있을 때, '1번지', '2번지', '3번지'라는 주소, 그리고 '덧셈'이라는 연산의 종류 등이 이 단계에서 확정된다. 또한 확정된 주소로부터 실행 단계에 필요한 값을 읽어 오기도 한다. 실행은 해당 명령어의 동작을 수행하는 단계로, 위 예시에서는 '덧셈'을 실제로 하는 과정이 이 단계에서 진행된다. 기억 장치 접근과 쓰기 단계는 동작상의 필요에 따라 각각 외부 기억 장치에 접근해서 데이터를 가져오거나, CPU 내의 기억 장치에 값을 갱신하는 단계이다.

㉮단일 사이클 방식 CPU로 모든 명령어를 처리하기 위해서는 실행하는 데 가장 오랜 시간이 걸리는 명령어를 기준으로 클럭 주기를 설정해야 한다. 가령, 외부 기억 장치로부터 값을 가져와 CPU 내 기억 장치에 저장하는 ㉠불러오기(LW) 명령은 명령어 실행의 다섯 단계를 모두 거쳐야만 정상적으로 수행할 수 있음이 알려져 있다. 그러나 위 예시의 덧셈과 같은 명령의 경우 기억 장치 접근 단계는 거칠 필요가 없음에도 불구하고, LW 명령과 같은 만큼의 시간을 소모하게 된다.

이러한 문제점을 해결하기 위해 등장한 것이 다중 사이클 방식 CPU이다. 다중 사이클 방식 CPU는 명령어 하나를 1 클럭 사이클에 처리하는 방법 대신 명령어 실행의 각 단계마다 1 클럭 사이클에 처리하는 방법을 사용한다. 예를 들어 LW 명령, ㉡덧셈 명령, LW 명령을 순차적으로 처리해야 하는 경우 14 클럭 사이클 만에 처리할 수 있다. 기본적으로 CPU는 수많은 명령어를 처리해야 하는데다가, ㉢처리하는 데 3단계로 충분한 명령도 있으므로 다중 사이클 방식 CPU는 단일 사이클 방식 CPU보다 처리 속도에 있어 훨씬 우월한 성능을 뽐낼 수 있다.

1. 윗글의 'CPI'와 '처리 속도'에 대한 이해로 적절하지 않은 것은?

① CPI가 일정하다면 '클럭 주기'가 짧은 CPU일수록 처리 속도가 빠르다.
② ㉠만을 수행할 수 있도록 설계된 단일 사이클 방식 CPU의 CPI와 ㉡, ㉢만을 수행할 수 있도록 설계된 단일 사이클 방식 CPU의 CPI는 서로 같다.
③ ㉠만을 수행할 수 있도록 설계된 다중 사이클 방식 CPU의 CPI와 ㉡, ㉢만을 수행할 수 있도록 설계된 다중 사이클 방식 CPU의 CPI는 서로 같다.
④ 2번의 ㉠과 3번의 ㉡만을 실행하는 프로그램을 다중 사이클 방식 CPU로 실행시켰을 때의 CPI는 4.4이다.
⑤ 다중 사이클 방식 CPU로 3번의 ㉠과 1번의 ㉢만을 실행하는 프로그램을 실행시켰을 때의 CPI보다 2번의 ㉠과 2번의 ㉡과 1번의 ㉢만을 실행하는 프로그램을 실행시켰을 때의 CPI가 더 작다.

2. 윗글을 바탕으로 <보기>를 이해한 반응으로 적절하지 않은 것은? [3점]

<보 기>

단일 사이클 방식 CPU의 각 단계를 담당하는 부분들을 인위적으로 분리하여 다중 사이클 방식 CPU처럼 작동하도록 만들 수 있다. 이런 CPU는 여러 개의 명령어를 동시에 수행할 수 있다. 즉 첫 번째 명령어가 해석 단계에 들어갔을 때 그다음 명령어의 인출 단계를 곧바로 수행하고, 앞의 두 명령어가 각각 실행과 해석 단계에 들어갔을 때 세 번째 명령어의 인출 단계를 곧바로 수행하는 것이다. 이를 파이프라이닝이라고 하는데, 이것을 마지막 명령어까지 반복할 수 있다면 CPU의 성능을 획기적으로 개선할 수 있다. 그러나 다중 사이클 방식 CPU에서는 나타나지 않던 오류들이 나타나기도 한다. 대표적으로 LW 명령 뒤에 덧셈 명령이 올 경우 쓰기 단계가 겹치게 되는 현상이 있는데, 이는 모든 덧셈 명령의 실행 과정에 기억 장치 접근 단계를 추가하고 그 단계 동안 아무 동작도 하지 않음으로서 회피할 수 있다.

① 파이프라이닝이 적용된 CPU로 ⓐ를 세 번 실행하는 데 7 클럭 사이클이 필요하겠군.
② ⓐ 바로 다음에 CPU 내 기억 장치 2번지의 값과 3번지의 값

을 더하여 3번지에 저장하라는 명령이 뒤따르는 프로그램은 파이프라이닝 CPU에서는 오류가 발생할 수 있겠군.

③ CPU 내 기억 장치 1번지의 값과 2번지의 값을 더하여 4번지에 저장하라는 명령 이후 ⓐ가 뒤따르는 프로그램은 파이프라이닝 CPU에서도 오류가 발생하지 않겠군.

④ CPU 내 기억 장치 1번지의 값과 3번지의 값을 더하여 2번지에 저장하라는 명령 이후 LW 명령과 ⓐ가 순서대로 뒤따르는 프로그램은 LW 명령이 4번지의 값만을 갱신한다면 파이프라이닝 CPU에서도 오류가 발생하지 않겠군.

⑤ 만약 프로그램 계수기의 값을 인위적으로 갱신하는 명령어가 프로그램에 포함되어 있다면 파이프라이닝 CPU에서는 오류가 발생할 수 있겠군.

3. ㉮의 이유로 적절한 것은?

① 실행 시간이 오래 걸리는 명령어일수록 프로그램에 자주 등장하기 때문이다.

② 프로그램 계수기의 값을 갱신할 시간이 필요하기 때문이다.

③ 다른 명령어에 비해 실행 단계가 특히 오래 걸리는 명령어가 존재하기 때문이다.

④ 외부 기억 장치에 접근하지 않더라도 CPU 내부 기억 장치에 접근하는 경우가 있을 수 있기 때문이다.

⑤ 명령어에 따라서 클럭 주기를 바꿀 수 없기 때문이다.

<답은 뒷장에 있습니다.>

* 확인 사항

○ 답안지의 해당란에 필요한 내용을 정확히 기입(표기)했는지 확인하시오.

[정답]
③④⑤

[대강 해설]

1번

① : 그러하다 (O)
② : 단일 사이클 CPU의 정의상 CPI는 반드시 1로 같다. (O)
③ : 전자는 5인 반면 후자는 4를 넘을 수 없다. (X)
④ : 1문단의 CPI의 정의에 따라 계산해보면 맞다. (O)
⑤ : 마찬가지로 해보면 전자가 4.5, 후자가 4.2가 나와서 맞다 (O)

2번

① : 파이프라이닝에서는 덧셈 명령이라도 기억장치접근 단계를 생략하지 않는다고 했다. 대략 이러한 모양이 된다.
(접근 == 기억장치접근. 2글자를 맞춰야 보기 좋아서.)
인출(1)/해석(1)/실행(1)/접근(1)/쓰기(1)
-----/인출(2)/해석(2)/실행(2)/접근(2)/쓰기(2)
-----/-----/인출(3)/해석(3)/실행(3)/접근(3)/쓰기(3)
7 클럭 사이클로 충분하다. (O)
② : ⓐ에서 “3번지에 저장하라”는 것은 쓰기 단계에서 일어나는 일이다. ⓐ 직후에 저게 오면, ⓐ는 쓰기 단계는커녕 아직 실행 단계에서 허우적거리는데 두 번째 명령어의 해석 단계 때문에 3번지 값을 읽어야 한다. 값이 갱신될 여지가 있는 위치에 아직 갱신하지 않은 값을 읽어오라는 꼴이므로 오류가 날 수 있다. (O)
③ : ②와는 다르게, 읽기만 하는 건 수백 번을 읽어도 문제없다. 읽기만 하는 위치에서 값이 바뀔 이유는 없기 때문이다. (O)
④ : LW 명령이 1, 2, 3번지를 안 건드린대서 아무 문제가 없을 것 같지만 그렇지 않다. ②에 대해서 생각해보면 “쓰기도 전에 읽기”가 문제가 될 수 있다는 것인데, 쓰기가 일어나는 쓰기 단계는 읽기가 일어나는 해석 단계보다 무려 세 단계나 뒤에 있다. 중간에 LW가 껴도, 최초의 명령은 아직 기억장치 접근 단계에서 헤매고 있다. 따라서 오류가 발생할 수 있다. (X)
⑤ : 프로그램 계수기는 다음 실행할 명령어를 가리키게 된다. 이에 따라서 줄줄이 명령어가 인출부터 하나씩 실행되기 시작했는데, 중간에 프로그램 계수기의 값이 인위적으로 변경되어야 한다면? 그 뒤로 실행되기 시작한 명령어들은 싹 정지해 버려야 한다. 따라서 오류가 발생할 수 있다. 참고로, 작위적인 상황이 아니라 이런 명령어가 실제로 있다. (O)
=> 사실 눈치가 빠르면 ③, ④중에 답이 되겠구나 싶기는 할 것.

3번

①②③ : 개소리 (X)
④ : 맞는 말이긴 한데, 그게 ㉮의 이유는 되지 못한다. 애초에 외부 기억 장치와 CPU 내 기억 장치(레지스터)는 건드리는 타이밍이 아예 다르다. (X)
⑤ : 1문단 극초반에 나온다. CPU같은 거 돌리려면 “일정한 간격”으로 클럭이라는 이름의 전기 신호를 받아야 한다. 지금 명령어가 LW가 아니라서 시간이 남아돈다고 해서, 님아 클럭 땡겨서 주시죠 할 수 있는 게 아니라는 말이다. (O)