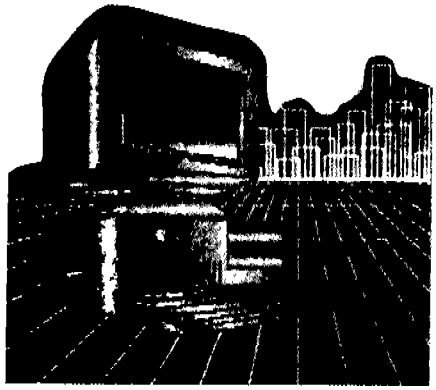
컴퓨터 메모리 구성요소의 동향



1

공학박사 이 근 철

제일전산훈련원 원장

◘ 서 론

컴퓨터 내부는 상당히 복잡하지만 이들을 구성하고 있는 소자를 크게 나누어 보면, 우선 독자들이 가장 잘 알고 있는 마이크로프로세서라고 부르는 CPU와 메인 보드에 정착되어 있는 여러 가지 소자들, 그리고 메모리에 관계된 소자 등으로 분류할 수 있다.

물론 이외에도 많은 부품들이 있지만 여기서는 가장 간단한 마이크로 시스템(Micro Computer System)을 기준으로 하여 각 소자의 역할에 대해 알아보기로 한다.

마이크로 컴퓨터 시스템은 컴퓨터의 가장 핵심이라 할 수 있는 마이크로프로세서 메모리 그리고 입출력 인터페이스가 있다.

마이크로프로세서는 메모리에 해당하는 램(RAM)이나 롬(ROM)에 저장되어 있는 데이터를 읽어 그 데이터를 명령에 따라 처리하는 부분이고, 메모리는 데이터를 저장하고 있는 장치이다.

시스템을 구성하고 있는 또 하나의 부분은 CPU가 외부기기와 데이터를 주고 받을 경우 서로 다른 방식으로 데이터를 전달할 때 이를 연결해 주는 입출력 인터페이스 부분이다.

최근에 학술잡지를 보면 세도우 램(Shadow RAM)은 무엇이며, 페이지 인터리빙 방식의 특수설계는 또 무엇인가, 또한 코프로세서(Coprocessor)를 부착하면 속도는 빨라지는가. 어째서 각 시스템마다 속도 차이가 나는가라는 기사를 볼 수 있다.

이번 호에서는 주기억장치인 메모리에 대하여 설명하고자 한다.

1. DRAM

DRAM(Dynamic RAM)이 내부에서 정보를 기억하는 방법은 콘덴서와 비슷한 역할을 하는 커패시터(Capacitor)의 특성을 이용하여 이루어진다. 이런 방법으로 정보를 기억하는 것은 원리상 매우 간단해서 높은 집적도의 메모리를 비교적 쉽게 만들 수 있지만, 커패시터에 충전된 전류는 어느 정도 시간이 흐르면 방전되어 원래 기억되어 있던 정보가 지워지는 치명적인 단점이 있다.

이 시간은 보통 4ms 정도로 DRAM이 원래의 정보를 잃어버리지 않고 계속 가지고 있기 위해서는 컴퓨터 회로에서 계속 재충전을 해주어야 하는데, 이러한 재충전 작용을 리프레시(Refresh)라고 한다. 또한 DRAM을 일정한 주기에 따라 리프레시 해주는데 걸리는 시간을 리프레시 주기(Refresh Cycle)라고 하는데, 이 주기마다 DRAM은 자기 자신의 데이터를 읽은 다음 재기입을 해준다. 물론 이렇게 리프레시가 이루어지고 있을 동안 그 메모리는 읽고 쓰는 일을 할 수 없다.

DRAM의 속도 규격에는 두 가지가 있다. 액세스 타임과 사이클 타임(Cycle Time)이 바로 그것인데, 우리가 보통 100나노, 120나노라고 말하는 것은 액세스 타임을 가리킨다.

CPU가 DRAM에 있는 정보를 읽으려 할 때는 우선 그 정보가 어디에 있는지를 가리키는 번지(Address)값을 DRAM에 준다. 그러나 번지값이 주어진 즉시 정보가 출력되어 나오는 것은 아니다.

어느 정도 시간이 경과되고 나서야 그 정보가 DRAM의 핀(pin)에 나타나게 되고, CPU가 정보를 가지고 갈 수 있는 것이다. 이 시간이 바로 액세스 타임이다.

흔히 사용되고 있는 12MHz급의 AT에는 100 ns 내외의 DRAM이 사용되고, 386PC에는 70ns 또는 80ns 정도의 액세스 타임을 가진 DRAM이 사용되고 있다.

사이클 타임은 액세스 타임과 재충전시간(Precharge Time)을 합한 것으로 메모리로부터 정보를 가져와 메모리를 정보로 쓰는데 실질적으로 걸리는 시간을 말한다.

재충전시간은 메모리에서 데이터를 한 번 읽고 난 다음 다시 읽을 수 있게 되기 전까지 기다려야 하는 시간을 말하는데, 이 시간은 액세스 타임만큼이나 길기 때문에 실제 DRAM의 사이클 타임은 액세스 타임의 약 두 배 정도가 된다.

예를 들어 액세스 타임이 80ns인 DRAM의 사이클 타임은 190ns이다. 즉, CPU는 매 190ns마다 DRAM에 정보를 쓰거나 읽을 수 있는 것이다. 마이크로 전자공학이 발달되어 가면서 DRAM의 집적도는 계속 증가되어 왔다.

아직도 기본적인 메모리 소자로는 256DRAM이 쓰이고 있지만 시장에는 1M DRAM뿐 아니라 4 M DRAM까지 판매되고 있고 조만간에 16M DRAM까지 출하될 전망이다.

그러나 DRAM은 이렇듯 용량면의 발전에 비해서 속도면에서는 그만큼의 발전에 못 미치고 있는 것이 사실이다. CPU의 속도가 빨라지면 빨라질수록 그에 비례해서 메모리의 속도도 빨라져야 시스템의 성능을 최대로 발휘할 수 있는 것이 당연한 일이지만, 물리적으로 DRAM의 속도가 CPU의 속도를 따라가지 못하기 때문에 이러한 속도문제를 해결하기 위해 일반적으로 대기상태(Wait State)를 이용하고 있다.

1·1 대기상태(Wait State)

메모리 속도의 증가방법으로 가장 일반적인 것이 바로 대기상태(Wait State)를 사용하는 방법인데, 이 방법은 메모리의 속도문제를 해결한 것은 아니고 단지 CPU가 메모리를 사용할 때 잠시동안 아무 것도 하지 않고 기다리는 것이다.

CPU는 밖에서 그 프로세스의 동작을 한 스텝만큼 지연시킬 때 사용되는 핀이 마련되어 있다. CPU의 외부에서 그 핀에 신호를 주면 CPU는 한 사이클을 대기주기(Wait Cycle)라고 말한다.

메모속도가 조금 느리다면 한 번의 대기상태(1-Wait State)로서도 충분히 CPU의 속도를 따라갈 수 있겠지만 그보다도 더 느리다면 두 번 혹은 그 이상의 대기상태가 필요하게 될 수도 있다.

대기상태는 이렇게 CPU를 아무 일도 하지 않고 기다리게 만들기 때문에 가능한한 대기상태를 가지지 않는, 즉 대기상태 0(Zero Wait State)의 동작을 하는 컴퓨터를 사용하는 것이 바람직하다. 얼마전까지도 PC에서는 대기상태가 있는 DRAM의 구조를 사용하는 것이 당연하게 여겨졌지만 메

모리 가격의 하락으로 요즘에는 AT의 경우에도 거의 대기상태 0으로 동작시키는 것이 보통이다.

그러나 바이어스 루틴이 들어있는 롬 바이어스의 경우에는 속도가 워낙 느리기 때문에 두 번의 대기상태(2-Wait State)를 가지게 하든지 세도우 램(Shadow RAM)이라는 기법을 사용하여 그 문제를 해결하고 있다.

AT의 경우에 단 한 번의 대기상태를 가지게 되면 대략 33% 이상의 속도 저하가 생긴다. 그래서 한 번의 대기상태를 가지는 16MHz의 시스템은 대기상태 0에서 12MHz의 클럭을 사용하는 PC와 비슷한 성능(속도)를 보이게 된다.

대기상태를 이용하면 어느 정도까지는 메모리의 속도를 늘릴 수 있겠지만 모든 단점을 극복한 완전한 대기상태 0의 동작을 하기 위해서는 SRAM의 사용이 아직까지는 가장 확실한 방법이라고 할 수 있다.

1·2 시스템 동작속도

최초의 IBM PC/XT는 시스템의 속도가 그렇게 빠르지 않았기 때문에 특별한 메모리 구조가 필요하지 않았다.

그 당시에는 속도보다는 오히려 메모리 용량이 더 중요한 요소였다. 그러나 AT가 발표되고 32비트 구조를 가진 386PC가 시장에 나타나면서 시스템의 성능을 높이기 위해 수정 발진기로부터 더 높은 클럭 주파수를 발생시켜 CPU가 더 빠른 속도로 동작하도록 처리하게 되었다.

그러나 클럭 주파수가 높다는 것은 단지 CPU의 내부에서만 그 속도로 명령어가 수행된다는 것을 의미할 뿐 전체 시스템이 항상 그 주파수로 동작하는 것을 의미하지는 않는다.

CPU가 칩 외부의 메모리나 주변장치 등과 서로 정보를 교환하려면 그 외부장치가 CPU에 맞는 속도로 액세스해 주어야 하며, 따라서 시스템의 속도는 CPU의 주파수 뿐만 아니라 그 시스템의 메모리가 어느 정도의 속도로 동작하느냐 하는 것도 아주 중요한 요소가 되는 것이다.

컴퓨터의 메모리로 쓰이는 칩은 롬(ROM)과 램(RAM)으로 구분할 수 있다. 롬은 보통 바이어스(BIOS) 루틴을 저장하는데 사용되고 그밖의 데이터나 응용 프로그램 등은 모두 램에 저장된다.

그러므로 롬보다는 램의 속도가 시스템의 성능에 있어서 더 중요한 변수가 된다. 시스템 설계자는 이러한 점을 감안하여 메모리가 CPU의 속도를 따라갈 수 있도록 해야 한다.

XT나 일반 AT 수준의 컴퓨터들은 흔히 사용되는 단순한 방식의 메모리 구조를 사용하여도 충분히 시스템의 운용이 가능하다.

그러나 보다 고성능의 AT나 386PC 정도의 수준에 이르게 되면 그러한 방식으로는 메모리가 더 이상 CPU의 성능을 따라가지 못하기 때문에 병목현상을 일으킨다.

이 현상은 PC의 클럭 주파수가 대략 16MHz를 넘으면서 발생하는데 이에 대응하기 위해 여러 가지 메모리 설계기법을 사용하게 된다.

램에는 SRAM(Static RAM)과 DRAM(Dynamic RAM)의 두 가지가 있다.

일반적으로 대부분의 PC들은 SRAM 대신 DRAM을 메인 메모리로 사용하고 있는데, 이것은 DRAM이 SRAM에 비해 값이 훨씬 싸기 때문이다.

SRAM은 속도가 빠르고 리프레시(Refresh)를 하지 않아도 되는 장점이 있지만 가격때문에 메인 메모리로는 거의 사용되지 않고, 80386을 CPU로 사용하는 PC에서 캐시용의 메모리로 사용되는 것이 보통이다.

2. SRAM(Static RAM)

SRAM에서는 정보를 기억하기 위해서 DRAM에서 사용되는 것과 같은 커패시터 방식을 쓰지 않고 플립 플롭(Flip-flop)의 구조를 사용한다.

플립 플롭은 한 번 정보를 넣어주면 전원이 꺼지기 전까지는 그 정보가 사라지지 않는다.

그러므로 전혀 리프레시 및 재충전(Precharge)을 해 줄 필요가 없고 따라서 이에 수반되는 속도 저하 및 복잡한 리프레시 회로 등 DRAM 사용시의 단점들이 해소된다.

그러나 이렇게 우수한 특성에도 불구하고 SRAM을 메인 메모리로 사용한 시스템이 거의 없는 것은 무엇보다도 가격이 높기 때문이다.

DRAM과 비교해 볼 때 한 바이트당 가격이 두 배 이상 가량된다. 또한 칩 내부의 구조가 복잡하기 때문에 동일한 용량일 때의 DRAM보다 4배 정도 칩 내부회로가 복잡해 칩의 크기도 더 커진다.

하지만 메인 메모리로 SRAM을 사용한 PC도 판매는 되고 있다.

표 I 에 SRAM과 DRAM을 비교해 놓았다.

현재까지 실용화되어 있는 기술로는 10ns 정도의 액세스 타임을 가진 SRAM까지도 만들 수 있으며, 10ns 이하의 것들도 발표되고 있다. 15ns 속도라면 대기상태 0에서 약 24MHz의 시스템 클럭을 가진 시스템에까지도 사용할 수 있다.

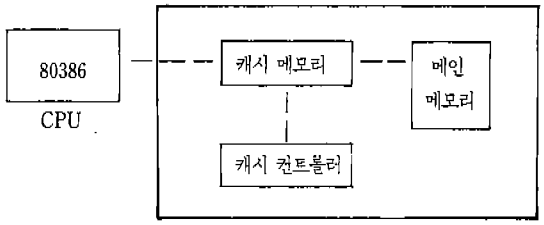
물론 속도가 빠르면 빠를수록 가격이 높아지기 때문에 이 정도 속도를 가진 SRAM들은 그리 많이 쓰이지는 않고 25ns 내지 35ns 정도의 SRAM들이 386PC의 캐시용으로 보통 사용된다.

3. 캐시 메모리

최근 들어서 386PC에서 흔히 사용되는 방법인 캐시 메모리(Cache Memory) 시스템은 가격대

<표 I > DRAM과 SRAM의 비교

구 분	DRAM	SRAM
리프레시및재충전	주 기 적	필 요 없 다
속 도	느 리 다	빠 르 다
회 로 구 조	단 순	복 잡
칩 의 크 기	작 다	크 다
가 격	저 가	고 가
용 도	일반메모리	캐시메모리



<그림 I > 386 PC의 캐시 시스템

성능비의 견지에서 보더라도 가장 바람직하고 확실한 메모리 속도문제의 해결방안이라고 볼 수 있다.

캐시 메모리 시스템에서는 메인 메모리로 가격이 저렴한 DRAM을 사용하고 속도가 빠른 SRAM을 프로세서와 메인 메모리 사이에 캐시로서 사용한다. 이것은 마치 속도 버퍼와 같은 형태가 된다(그림 I 참조).

캐시 메모리를 액세스 할 때 메인 메모리의 정보는 CPU 뿐만 아니라 캐시 메모리에도 저장된다.

이때 캐시 메모리는 그 정보가 있는 메인 메모리의 번지로부터 캐시 메모리 용량만큼의 정보를 계속 읽어들인다.

다음 번에 다시 CPU가 메인 메모리상의 정보를 가져오고자 할 때 캐시 제어회로는 그 정보가 캐시에도 들어있는 것인지 검사하여 캐시에 들어있으면 CPU가 캐시로부터 그 정보를 가져가게 만든다.

이때 물론 프로세서는 SRAM의 빠른 속도로 정보를 가져간다.

만약 그 정보가 캐시에 들어있지 않다면 캐시 제어회로는 메인 메모리로부터 그 정보를 가져오면서 동시에 다시 캐시에도 로드한다.

이런 식으로 수행하다 보면 다시 캐시에 있는 데이터를 CPU가 액세스하는 양상이 된다.

물론 모든 캐시 회로가 이런 식으로 동작하는 것은 아니다. 캐시가 데이터를 메인 메모리로부터 한 블럭만큼 로드해 올 동안 CPU가 멈춰 있게 만들

어진 시스템도 있다.

이런 것은 캐시 회로가 어떤 알고리즘을 사용하는가에 따라 달라진다. 이때 CPU가 액세스하고자 하는 정보가 캐시에 있을 확률을 적중률(Bit-rate)이라고 하며 없을 확률을 실패율(Miss-rate)이라고 한다.

캐시의 사이즈가 크면 클수록, 즉 16KB보다는 32KB가, 그 보다는 64KB의 캐시 시스템에서 더 적중률이 커지게 되겠지만 실패시의 경우에는 메인 메모리에서 로드하게 되는 시간이 캐시의 사이즈가 커질수록 길어지기 때문에 캐시의 크기를 무조건 늘리는 것이 좋은 것만은 아니다.

표 2는 80386 CPU를 사용한 시스템에서 캐시를 사용했을 때와 사용하지 않았을 때를 비교한 자료로서 캐시의 사이즈와 방식 등에 따른 적중률을 보여 준다.

이 표에 의하면 1KB 크기의 캐시를 사용했을 때는 오히려 사용하지 않는 것보다 못함을 알 수 있고, 캐시의 사이즈가 어느 정도 커지면 더 이상 적중률이 올라가지 않음을 알 수 있다. 여기서 적중률은 최고 93%까지로 나타난다.

캐시 시스템의 착안원리는 CPU가 메인 메모리를 액세스 할 때 대체로 어떤 한 영역에 집중되는 사실에 근거한다. 서브루틴을 콜하거나 인터럽트가 걸리는 경우 등을 제외하고는 보통 어떤 특정 루프를 돌게 되면서 동일한 메모리 영역들을 계속하여 액세스하는데 이것을 지역성(Locality)이라고 한다.

캐시 시스템을 운영하기 위해서는 상당히 정교하고 복잡한 제어회로가 필요하다. 따라서 캐시 제어용의 전용 컨트롤러 칩을 사용하게 되는데, 대개의 경우를 보면 인텔의 82385 칩을 사용하는 것이 보통이고 C & T(Chips & Technology)사의 82C307을 사용하는 시스템도 있다.

82385를 사용하게 되면 Direct-mapped 방식과 Two-way Set Associative 방식 가운데 한 가지 방법으로 주로 운용된다.

캐시가 일반적인 프로그램을 수행할 때는 제 성능을 발휘할 수 있지만 어떤 경우에는 무용지물이 되거나 오히려 시스템의 능률을 떨어뜨릴 수도 있다.

가령 386 PC에서 멀티태스킹(Multi-tasking)을 한다고 해보자. 그렇다면 시스템은 각 프로세서간을 순회하게 된다. 따라서 제어가 새로운 프로세서를 옮겨갈 때마다 캐시가 메인 메모리로부터 다시 로드되므로 로드시간이 너무 많이 걸리게 된다. 입출력 명령을 많이 수행하는 프로그램의 경우도 비슷한 결과가 된다.

또 한 가지는 프로그램에서 코드 영역과 데이터 영역이 따로 존재하면서 데이터를 많이 액세스하는 프로그램을 돌 수 있다.

이때 캐시는 계속 두 영역 사이를 오가면서 메인 메모리 로드를 되풀이하게 될 것이고 성능은 크게 떨어질 것이다.

그러나 이같은 문제점이 존재함에도 불구하고 역시 캐시 시스템은 386 PC의 성능을 향상시키기 위한 가장 앞선 기법임에 틀림없고 이 기법을 사용하는 386 PC도 계속 늘어갈 것이다.

☞ 다음 호에 계속

<표 2 > 캐시의 형태에 따른 성능비교

캐시의 형태			캐시의 성능	
캐시용량 (바이트)	캐시 형태	라인크기 (바이트)	적중률 (%)	캐시를 사용하지 않은 상태의 성능비
1K	Direct-mapped	4	41	0.91
8K	Direct-mapped	4	73	1.25
16K	Direct-mapped	4	81	1.35
32K	Direct-mapped	4	86	1.38
32K	Two-way	4	87	1.39
32K	Direct-mapped	8	91	1.41
64K	Direct-mapped	4	88	1.39
64K	Two-way	4	89	1.40
64K	Two-way	4	89	1.40
64K	Direct-mapped	8	92	1.42
64K	Two-way	8	93	1.42
128K	Direct-mapped	4	89	1.39
128K	Two-way	4	89	1.40
128K	Direct-mapped	8	93	1.42