Introduction to Computer Systems

알기쉬운

電子情報處理組織(EDPS) ◀ Ⅳ ▶

圖協出版部

CHAPTER 4.

중앙처리장치(CPU)

4.1 개 요

중앙 처리 장치는 전체의 컴퓨터 시시템을 제어하고 관리하며, 데이터에 대한 실제적인 연산 및 논리 작업을 수행한다. 기능적인 측면에서 볼 때, CPU는 제어(control)와 연산/논리(arithmetic/logical)의 두 기능으로 구성되어 있다(Fig. 4-1).

제어 기능은 명령어에 의해 불려지는 모든 조작을 감독하며 조정한다. 이 제어 기능은 입출력 장치의 제어, 기억장치로부터 정보의 입력과 이동, 기억 장치와 연산논리 기능 사이에서 정보의 순차 지령 등을 포함한다. 제어 장치의 활동을 통하여, 전체 컴퓨터 시스템의 자동적이고 통합된 조작이 이루어진다.

여러가지 측면으로 볼 때 제어 장치는 전화교환대와 비유될 수 있다. 모든 가능한 데이터가 있는 것과 꼭 같이 중앙 교환소에 의해 공급된 모든 전화들 사이에 연결선이 이미 만들어진 통로로 전송될 수 있다(Fig. 4-2).

전화 교환대는 음성 펄스를 한 전화로부터 다른 전화로 옮기고, 전화를 울리고, 회로를 연결하고, 차단하는 등 전화를 제어하는 수단을 가지고 있다. 한 전화와 다른 전화 사이의 대화 통로는 교환대 자체의 적절한 제어에 의해 이루어지게 된다. 컴퓨터에 있어서, 명령어의 실행은 주어진 조작을 위해 많은 통로나 Gate의 개폐를 동반한다. 제어장치의 몇 가지 기능은 입출력 장치를 작동시키거나 정지시키고, 신호장치를 켜거나 끄며, Tape reel을 다시 감고, 연산의 과정을 감득하는 것이다. System/360 모델에 있어서, 이제어 장치의 한 부분은 Operation code에 의해 지적되는 조작을 수행하기 위한 회로를 가진 Read-only-storage라고 불리는 제어 장치로 구성되어 있다. 이 제어 장치는 다른 컴퓨터를 위해 쓰여진 프로그래밍 명령어를 System/360이 수행하도록 하기 위해 사용자가 선택할 수 있는 Emulator circuit를 넣어 둔다.

연산논리 장치는 연산과 논리 조작을 수행하기 위한 회로를 가지고 있다. 연산 부분은 연산하고, 수를 옮

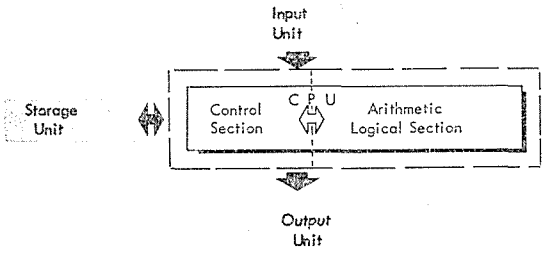

Fig. 4-1. Central processing unit in the data processing system

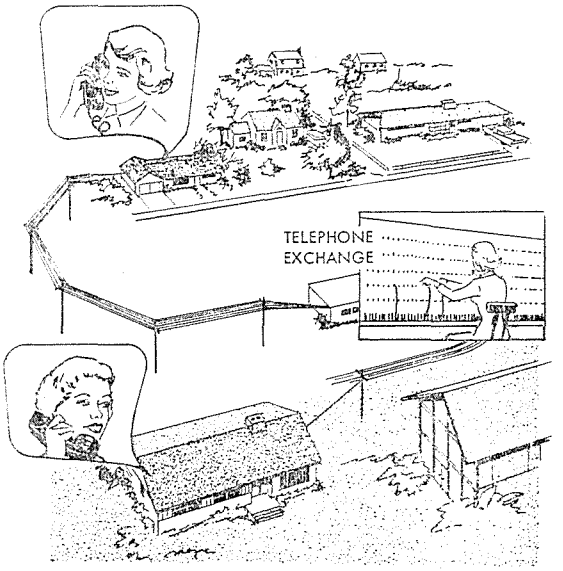

Fig. 4-2. Telephone exchange system

기고(shift) 연산 결과의 대수적인 부호를 붙이며, 반올림 하거나 비교 등을 하며, 논리 부분은 명령어 처리의 순서를 바꾸기 위한 Decision-making 조작을 수행한다.

4.2 기능별 장치(Functional Uunit)

레지스터(Register)

Register는 정보를 받아들이고, 그것을 보관하고 있다가 제어 회로의 지시로서 직접적으로 정보를 전달할 수 있는 장치이다. 사용되는 전자적인 구성 요소로서 마그네틱 코어나 트랜지스터를 사용할 수 있다.

Register는 그들의 기능에 따라 이름이 부여된다. 즉 Accumulator는 결과를 모으며, Multiplier-quotient는 곱함수(multiplier)와 상(quotient)을 보관하며, 기억 레지스터(storage register)는 정보를 Storage에서 보내주거나 그곳으로부터 가져온 정보를 갖고 있으며, Address register는 기억 장소나 장치의 번지를 보관하며, Instruction register는 처리될 명령어를 보관한다(Fig. 4-3). System/360은 범용 레지스터를 가지고 있는데, 이것은 Storage addresses, Index addresses, 논리적 또는 연산적으로 처리되기 위한 데이터를 포함하는 몇가지 기능을 위하여 사용된다.

레지스터는 크기, 용량, 용도에 있어서 차이가 있다. 몇가지 경우에 있어서, Extra position은 연산을 하는 동안에 가능한 Overflow condition을 찾아낸다.

예를 들면, 만약 두 개의 11자리 숫자를 더하면, 그 결과는 12자리의 해답이 될 수 있다(Fig. 4-4).

〔Fig. 4-4〕에서 Register A는 한 인수를 가지며. Register B는 다른 인수를 갖는다. 이 두 개의 인수가 결합되어지면, 그 결과는 Overflow position에 데이터로 나타남으로써, Overflow condition이 표시되는 C register에 놓여지게 된다. 다른 Register의 내용은 그 Register 안에서 좌측이나 우측으로 옮겨지며, 어떤 경우에는 Register들 사이에서도 행해진다. 〔Fig. 4-5〕는

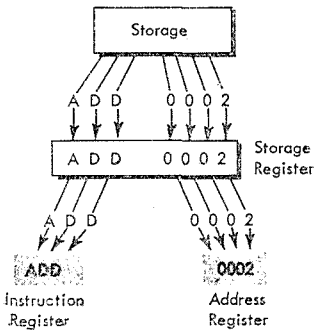

Fig. 4-3. Register nomenclature and function

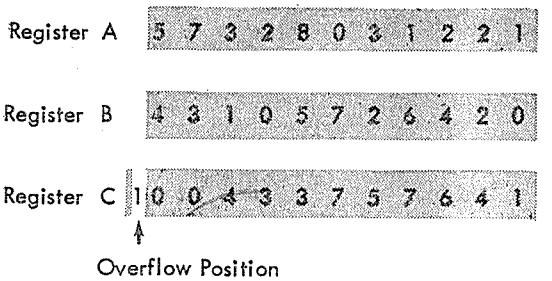

Fig. 4-4. Overflow condition resulting from addition

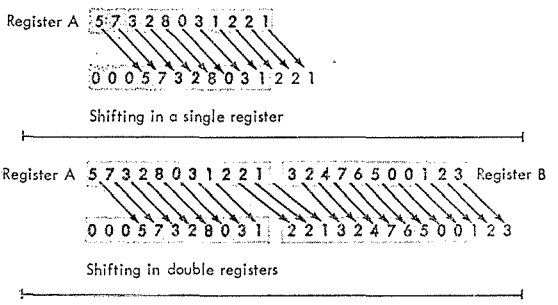

Fig. 4-5. Types of computer register shifting

3 position 만큼 우측으로 Register의 내용을 옮기는 것을 보여준다. 비어 있는 자리는 0(zero)으로 채워지며, Register 용량을 넘쳐 옮겨진 숫자는 변하게 된다.

다른 경우에 Register는 구성된 회로가 데이터를 분석하는 동안 데이터를 보관한다. 예를 들면, 한 개의 명령어(instruction)는 Register 내에 놓여 질 수 있으며, 구성된 회로는 연산을 실행할 수 있으며, 사용될 데이터의 위치를 배치한다. 특정한 Register 내에 있는 데이터는 역시 타당성을 검사 받게 된다.

한 시스템에 있어서 아주 중요한, 특히 정상적인 데이터의 흐름과 Storage Addressing을 포함하는 Register들은 그들과 관련 있는 작은 전등을 가지고 있다.

이 전등은 Register의 내용과 여러가지 프로그램 상태를 눈으로 표시하기 위하여 〔Fig. 4-6〕과 같은 콘솔(console) 위에 자리를 잡고 있다.

카운터(Counter)

Counter는 Register와 비슷하며 몇 가지 같은 기능을 수행할 수도 있다. 그 내용은 증가될 수도 감소될 수도 있다. Counter의 작동은 컴퓨터 시스템 내에서 그 설계와 용도에 관계가 있다.

Counter는 Register와 마찬가지로 Machine console 위에 눈으로 볼 수 있게 표시를 할 수 있다.

더하기 장치(Adder)

Adder는 데이터를 두 개 이상의 Source로부터 받아들여서 덧셈을 하고, 그 결과를 받아들이는 Register나

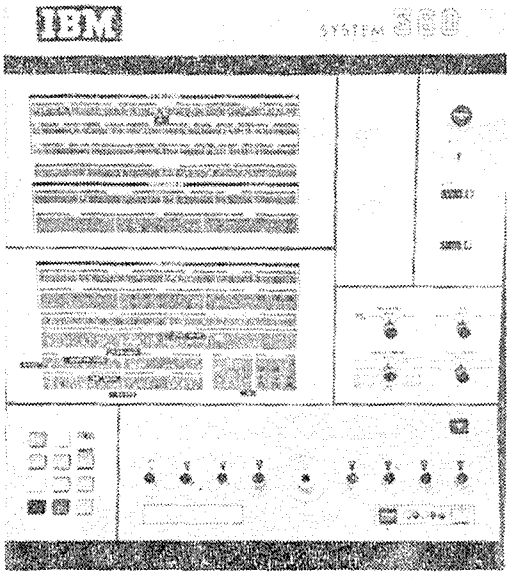

Fig. 4-6. Typical operator console

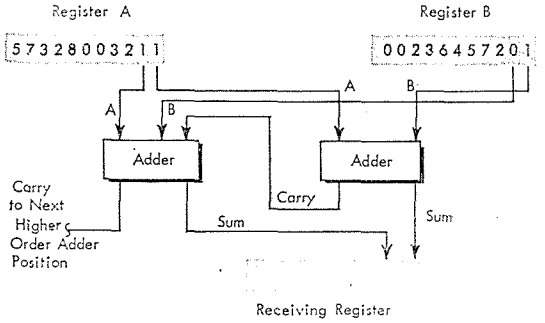

Fig. 4-7. Adders in a computer system

Accumulator에 보낸다. [Fig. 4-7]은 Register A와 B로부터 입력되는 가산 회로의 두 위치를 보여 준다. 합계는 Adder에서 나타난다. 어떤 위치에서 오는 캐리(carry)는 Next-higher-order position으로 보내어진다. 마지막 합계는 받아들이는 레지스터(receiving register)의 관계 있는 위치로 옮겨지게 된다.

4.3 기계의 작동(Machine Cycle)

모든 컴퓨터 작동은 고정된 시간 간격으로 일어난다. 이러한 간격(interval)은 전자 시계로부터 5백만분의 1초 단위로 발생하는 규칙적인 펄스에 의해 측정된다. 펄스(pulse)의 수는 각각의 기본적인 Machine cycle의 시간을 결정한다.

컴퓨터 어법에 있어서 시간 관계는 천분의 1초, 100만분의 1초, 10억분의 1초와 같은 말로서 나타낸다. 이러한 용어는 천분의 1초가 얼마나 짧은 시간 간격(time interval)인가를 깨닫지 않고서는 아무런 의미도 전할 수 없다.

예를 들면, 눈을 한번 깜짝이는데 1000분의 1에서 2초 또는 수백분의 1초가 걸린다.

아래의 도표는 몇가지 부가적인 용어와 생략형을 나타낸다.

. 1=1/10초
  =100 milliseconds
. 001=1/1000초
  =1 millisecond(msec)
. 00001=1/1,000,000초
  =1 microsecond(μsec)
. 000000001=1/1,000,000,000초
  =1 nanosecond(ns)

Machine cycle 동안, 컴퓨터는 특정한 Machine operation을 수행할 수 있다. 하나의 명령어를 처리하는데 필요한 많은 조작은 그 명령어에 달려 있다. 여러가지 기계 조작은 각 명령어를 실행하기 위하여 이와 같이 결합된다.

명령어는 보통 최소한 Operation과 Operand의 두 부분으로 구성되어 있다. Operation은 어떠한 기능, 즉 read, write, add, subtract 등을 수행할 것인지를 기계에 알려주고, Operand는 주기억 장치 내의 데이터나 명령어의 번지, 보조기억 장치 내의 데이터나 프로그램의 번지 또는 입출력 장치의 번지이다. 이것은 역시 Register 내에서 양을 Shifting하는 것과 또는 테이프의 Reel을 Backspacing과 Rewinding 같은 제어기능을 지시할 수도 있다.

명령어를 받아들이고, 번역하고, 실행하기 위해서, 중앙처리 장치는 특정한 명령어에 의해서 결정·지시된 순서대로 수행되어야 하며, 그 명령어는 Time pulse의 Fixed interval 동안 수행되어야 한다.

명령어의 작동(Instruction Cycle)

명령어를 실행하는데 필요한 첫 Machine cycle을 명령어 사이클이라 부른다. 이 cycle에 대한 필요한 시간은 명령어 수행 시간이다. 명령어의 수행 시간 동안

1. 명령어의 주기억 장치의 Location으로부터 불려서 중앙처리 장치로 옮겨진다.
2. Operation 부분은 Instruction register에서 코우드화 된다. 이것은 기계에게 어떠한 조작을 행할 것인가를 알린다.
3. Operand는 Address register에 위치하게 되고 이것은 기계에게 조작에 사용될 요소들을 알린다.
4. 수행된 다음 명령어의 Location이 결정된다.

프로그램의 시초에, Instruction counter는 첫번째 명

령어의 번지로 세트된다. 이 명령어는 주기억 장치로부터 옮겨지며, 이것이 수행될 동안 Instruction counter는 다음 명령어가 들어 있는 번지수로 자동적으로 당겨진다.

만약, 각 명령어가 하나의 기억 장소를 차지하게 되면 Counter는 1이 증가되고, 만약 5개의 기억 장소를 차지하면 5가 증가된다. 하나의 명령어가 수행될 때까지 Counter는 프로그램 순서에서 나타나는 다음 명령어의 번지로 정해진다. Counter가 증가하는 것은 자동적이다. 다른 말로 바꾸면 컴퓨터가 일련의 명령어의 연결로 수행하여 나갈 때 Counter는 명령어에서 다른 방식으로 수행하도록 할 때까지 차례대로 명령어를 수행한다.

누적 레지스터(accumulator register)의 내용에 기억장소 2의 내용을 더하라는 명령이 주어졌다고 가정하자. [Fig. 4-8]은 관련된 주 Register와 정보의 유통선을 나타낸다.

I-time은 Instruction counter가 명령어의 기억장소를 Address register에 옮겨줄 때 시작된다. 이 명령어는 주기억 장치로부터 선택되어서 Storage register에 놓이게 된다. 기억 레지스터(storage register)로부터의 Operation part는 Instruction register로, Operand는 Address register로 보내진다. Operation decoder는 명령어를 수행하기 위해 특정한 회로 통로를 정한다.

명령어의 수행에서 순차적인 진행은 필요하지 않다. 어떤 명령어는 순차적인 수행의 과정을 무제한으로 바꾼다. 이러한 경우에, 기억 장치로부터 옮겨지는 명령어는 다음의 순차적인 명령어로 처리되지 않고, 다른 위치에 있던 것이 다음에 올 것을 지정한다.

Instruction counter가 변하는 단계는 전체 프로그램이 다음에 오는 데이터의 그룹을 반복할 수 있도록 하기 위하여 프로그램의 시작을 뒤로 재배치할 수 있다.

선택할 수 있는 명령어를 전송하는 이러한 Branching (transfer)은 역시 조건부라고 할 수 있고, 컴퓨터는 어떤 지정한 장치를 조사하기 위하여 직접 사용되며, 뒤

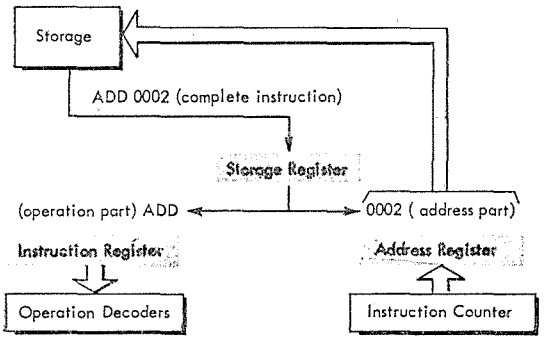

Fig. 4-8. Computer I-cycle flow lines

에 Indicator가 켜졌는지, 꺼졌는지에 따라 Branch한다. 사실 이러한 명령어는 Accumulator 내에 있는 양의 부호를 보라. 만약, 부호가 음수이면 5000 location으로부터 다음 명령어를 가지며, 부호가 양수이면 순서대로 다음 명령어를 진행하라고 할 수 있다. Instruction counter는 두 가지 가능한 기억 장소(5000이나 순서에 따른 다음 명령어의 장소) 중의 하나에 따라 지정된다. 컴퓨터가 따르는 논리적인 통로(즉, 처리되는 명령어의 정확한 순서)는 무제한의 Branching이나 여러 관점에서 적용되는 일련의 조건적인 테스트에 의해 제어될 수 있다. 그렇지만 기억 장치 내에서 명령어의 배열은 정상적으로는 변하지 않는다.

실행 사이클(Execution Cycle)

실행과 E-time 동안 일어나는 하나나 그 이상의 Machine cycle은 보통 I-time을 따른다. 필요한 많은 실행 사이클의 수는 수행될 명령에 따르게 된다.

[Fig. 4-9]는 [Fig. 4-8]에서 설명된 I-time에 따르는 데이터의 흐름을 보여준다.

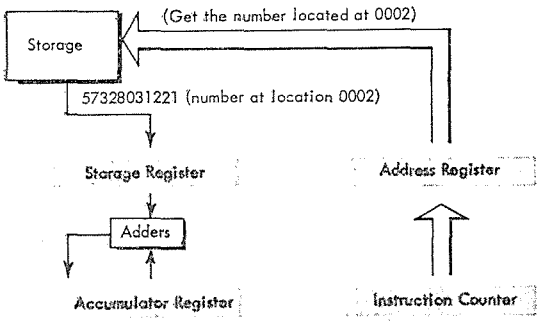

Fig. 4-9. Computer E-cycle following an I-cycle

실행 사이클은 Address register에 의해 지정된 번지(0002)에 위치한 정보를 기억 장치에서 옮기는 것에 의해 시작한다. 이 정보는 Storage register에서 자리를 잡게 된다. 이 경우에, 더해질 요소들 중의 하나는 Accumulator로부터 온 수와 함께 Adder 속에 자리를 잡게 된다. Storage register와 Accumulator의 내용은 Adder 내에서 결합되며, 그 합계는 Accumulator로 되돌아 가게 된다.

Address register는 데이터의 기억장소와 다른 정보를 갖고 있다. 이것은 입출력장치의 번지와 또는 수행될 제어기능을 지정한다. 명령어의 Operation part는 어떻게 이 정보를 번역하는가를 기계에 말하게 된다.

4.4 Serial and Parallel Operation

컴퓨터와 컴퓨터의 부분은 연산을 수행하는 방법에 따라 연속적인 것과 병렬적인 것으로 분류된다. 본질

	1st Step	2nd Step	3rd Step	4th Step
Addend	1234	1234	1234	1234
Augend	2459	2459	2459	2459
Carry	1	1		
Sum	3	93	693	3693

Fig. 4-10. Serial addition

적으로, 모든 연산은 덧셈으로 수행된다.

Serial 컴퓨터에 있어서, 더해지는 수는 덧셈이 종이와 연필로서 행해지는 것과 같은 방법으로 동시에(100등) One position으로 생각된다. Carry가 발생할 때마다 이것은 일시적으로 기억되며, 그 때 그 다음의 윗자리의 합에 더해진다. 연속적인 조작에 필요한 시간은 더해질 요소들의 자리수에 따르게 된다. 연속적인 덧셈은 [Fig. 4-10]에서 보여준다.

Parallel computer에 있어서, 덧셈은 완전한 Data word 로서 수행된다. Word는 Carry를 포함한 하나의 조작으로 결합된다. Word에 포함된 수의 크기에 관계없이 어떤 두 개의 Data word는 동시에 더해질 수 있다. [Fig. 4-11]은 Parallel addition을 보여준다.

	00564213
Numbers being added	00000824
Carry	1
Final Result	00565037

Fig. 4-11. Parallel addition

Fixed-length and Variable-length Word

컴퓨터는 Fixed-length나 Variable-length word를 사용하여 데이터의 번지를 지정하거나 처리할 수 있다.

Fixed-length word를 사용하는 연산에 있어서, 정보는 미리 정해진 Position의 수를 가진 단위나 Word로서 취급되거나 번지가 정해지게 된다. Word의 크기는 시스템 내부에서 설계되며, 이것은 보통 중앙처리 장치에서 처리하기 위하여 번지를 지정할 수 있는 정보의 최소 단위와 일치한다. Record, Field, Character, Factor는 모두 Parallel에서 Word로 다루어질 수 있다. Register, Counter, Accumulator, Storage는 표준 Word에 적응될 수 있게 설계되었다.

Variable-length word를 사용하는 조작에 있어서, Data-handling 회로는 단일한 Character로서 정보를 연속적으로 처리하기 위해서 설계되었다. Record, Field, Factor는 기억 장치의 용량 안에서 어떤 실질적인 길이일 것이다. 정보는 Word 대신에 Character가 유용하다. 주어진 자료처리 조직 내에서의 연산은 모두 Fixedlength거나 모두 Variable 또는 이 두 가지의 결합이다.

4.5 Floating-point Operation

수학자나 과학자들은 아주 큰 수와 아주 작은 유효숫자(fraction)들을 간단히 하도록 대수(logarithm)를 사용한다. 같은 이유로 과학용 연산을 위해서 Computer에서는 Floating-point 연산을 사용하며, 그리고 다양한 응용을 위한 Time-share computer의 출현으로 다목적 Computer에서 Floating-point는 일반화되었다.

우선 Floating-point를 사용하는 7090/7094 D.P. System의 원리를 고찰해 보자. 기본적인 원리는 사무용 Computer이고, 결국 System/360에 다목적으로 적용된다. Floating-point 연산을 하는 모든 중앙처리 장치들은 수를 변환하여 지수의 형태로 방정식의 오른쪽에 나타낸다. 예를 들면,

$$N=b^e \times f$$

여기서

N……수(number)

b……base(2진수에 대해서 2, 10진수에 대해서 10과 같이)

e……지수

f……fraction

Base는 기억장치 내부의 진법에 따라서 사용되며, 또한 Base는 computer에 의해서 Fixed 되기 때문에 7090/7094에서는 2이고, System/360에서는 16이다. 그리고 Base는 공식으로부터 삭제되며, 그 때 읽기를

N=exponent×fraction

Fixed-length word를 사용하는 Computer들은 Space에 따라 사용할 수 있는 지수의 허용 범위가 정해지거나, Computer에서 Position에 따라 사용할 수 있는 범위가 지수에 대한 sign(+or−)이 포함되어 결정된다.

Space 할당은 2^{127}(decimal exponent)의 Binary range가 허용되어 System/360에서 Sign bit 없이 8 bit이다. 이들의 값은 대략 $10^{38}\sim10^{-38}$이다.

System/360에서 Sign bit가 없는 7 bit는 $16^{63}\sim16^{-64}$의 Hexadecimal range가 된다. 이것은 10진수로 변환되고 이 값은 대략 $10^{76}\sim10^{-77}$ 범위이다.

Sign bit가 없는 Position은 지수에 허용될 때, 가능한 지수에 전체 범위의 중앙 값은 지수 0을 대표하기 위하여 선택된다. 양의 지수들은 그 값이 'Middle'에 더해진다. 음의 지수들은 그 값이 'Middle'에서 빼진다. 'Scaled exponent'는 Characteristic이라고 부른다.

Fixed-length word를 사용하는 Computer에서 Fraction의 범위는 지수나 Sign bit가 준비된 후에 Word (또는 words)에 남은 Space에 의해 결정되고, Floating-point 수를 표현하기 위해 할당된다. System/360에서

하나 또는 둘의 32-bit word를 Fraction의 자리가 사용됨에 따라서 Fraction field는 6 또는 24 hexadecimal digit(24 or 56 bit)까지 차지할 수 있다. Fixed-length word를 사용하는 다른 Computer의 중앙처리 장치들은 언제나 한 Word 또는 두 Word의 수를 기억하도록 지정함에 따라 Single 또는 Double-precision이라 부르는 두 개를 교대로 취한다. 7090/7094에서 Single-precision fraction은 27 bit 길이이고, 7070/7074에서는 8 bit 길이까지이다.

Fixed-length word를 갖지 않은 Computer에서 Floating-point fraction들의 길이는 :

IBM 1410 −2부터 97 digit

IBM 1710 −2부터 18 digit

수의 표현 방법으로 Exponent와 Fraction(유효 숫자)을 사용하기 때문에 Floating point라고 부른다. 우리는 지수로 변경할 수 있고, 그리고 그 때 Decimal point(decimal computer에), Binary point(binary computer)에서, Hexadecimal point(hexadecimal computer에서)로 일치시켜 'FLOAT'로 변경할 수 있지만, [Fig. 4-12]와 같이 수의 값은 변경할 수 없다.

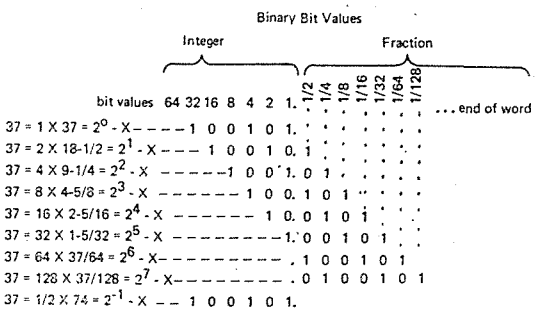

Fig. 4-12. Different floating-point expressions of the same value

Point가 High-order 1 bit(binary로)와 Fraction의 High-order significant digit(decimal과 hexadecimal로)의 왼쪽으로 올 때, Floating-point 표현은 표준화 되었다고 부른다. 수들이 최초로 Floating-point 수로 Computer의 중앙 처리 장치 속으로 들어 갈 때, 수들은 표준적인 형태를 갖게 된다. 연산 조작이 이 방법으로 될 때, Point는 수들을 2의 같은 Power까지 얻기 위해 찾을 때, 오른쪽 또는 왼쪽으로 옮겨질 수 있다. 그 때 수(number)들은 명확하게 비표준으로 된다. 그러나 경우에 따라서는 Computer는 마지막의 결과가 표준화되어질 수도 있다.

Binary Floating-point Notation

지수들의 Suject나 Binary computer에서 특성이 진행되기 전에 Clear되어야 한다.

	Decimal Exponent	Decimal Characteristic	Binary Characteristic
$10^{38} \approx 2^{127}$	127	255	11 111 111
$10^{0} \approx 2^{0}$	0	128	10 000 000
$10^{-38} \approx 2^{-128}$	128	0	00 000 000

Fig. 4-13. Range of characteristics in an IBM 1130 Computing system

Dec. No.	Decimal Representation Exponent	Fraction	Characteristic Bit Positions	Fraction Bit Positions
$1 = 2^1 \times 1/2$	1	1/2	10 000 001	100 000 000
$2 = 2^2 \times 1/2$	2	1/2	10 000 010	100 000 000
$3 = 2^2 \times 3/4$	2	1/2+1/4	10 000 010	110 000 000
$4 = 2^3 \times 1/2$	3	1/2	10 000 011	100 000 000
$5 = 2^3 \times 5/8$	3	1/2+0/4+1/8	10 000 011	101 000 000
$10 = 2^4 \times 5/8$	4	1/2+0/4+1/8	10 000 100	101 000 000
$.5 = 2^0 \times 1/2$	0	1/2	10 000 000	100 000 000
$.005 = 2^{-2} \times 1/2$	-2	1/2	01 111 110	100 000 000

Fig. 4-14. Examplex of normalized binary floating-point numbers

Characteristic들은 Sign이 없다. 그러므로, 앞에서 서술한 것과 같이 0(zero) 지수는 Characteristic field (8 bit)에서 가능한 값의 범위가 중앙에 놓이게 된다. Characteristic과 유효 숫자(fraction)가 항상 Octal로 표현되므로 쉽게 이해하기 위해서 2진수로 바꾸어진다. [Fig. 4-13]은 가장 큰 양의 Decimal-octal 값 0(zero)과 가장 큰 음의 지수, 그리고 8진수, 2진수 특성에 일치되는 관계를 보여 주고 있다.

Floating-point 수로 표준화된 2진수 System에서는, Fraction은 언제나 High-order binary 1을 가지고 있다. 그러므로 Fraction은 항상 같거나, 보다 크거나, $\frac{1}{2}$, 1보다 작거나 한다. [Fig. 4-14]는 간단한 10진수를 표준화된 Binary floating-point로 변환하는 방법을 보여 주고 있다.

10진수의 Integer들을 2의 지수와 Fraction의 결합으로 변환(convert)하는 쉬운 방법은 다음과 같다.

1. 10진수를 8진수로 변환(참고 8진법). 예 37에 대한 8진수는 45이다.
2. 8진수를 2진수로 변환(45에 대한 binary는 100101이다).
3. Binary point(현재 binary에서 말하므로 소수점이라 할 수 없다)를 2진수(여기서는 100101)의 오른쪽에 놓아라.
4. Binary point를 2진수(100101)에서 제일 왼쪽으로 옮겨라.
5. Binary point는 [Fig. 4-14]에서 수직선이 있는 곳이다. 2진수를 그 선 100101의 오른쪽으로 바로 놓아라. 이것은 1/2과 1 사이에 하나의 Fraction이다.

6. Binary point가 왼쪽 6 digit로 옮길 때 지나간 Binary digit의 수는 Count된다. 그 수는 2의 정확한 지수이다.
7. Characteristic 범위의 중간 지점(midpoint)의 수를 더하고 그것을 수직선의 왼쪽에 넣어라.

예 :

characteristic	fraction	의미
1000 0110	100101	

$2^6 \times 37/64$ or $64 \times 37/64 = 37$

10진법 Fraction으로부터 2의 지수와 Fraction($\geq \frac{1}{2}$ 또는 <1)으로 변환은 우리가 Binary point를 왼쪽 대신에 오른쪽으로 옮기고, 그리고 처리에서 2의 음수 지수를 증가시키는 것을 제외하고 유사한 과정이다.

1. Decimal fraction을 8진법으로 변환한다. 예로서 10진법 149는 8진법으로 $.1142^{+}$ 이다.
2. 8진법을 2진법으로 변환한다(8진법 1142의 2진법은 001 001 100 010이다).
3. Binary point를 2진수 왼쪽에 놓는다(.001 001 100 010).
4. Binary point를 Binary fraction에서 가장 왼쪽의 왼쪽으로 옮김으로 오른편으로 이동된다(00.100 110 001).
5. Binary point(two digit)의 Shifting에서 지나간(pass over) Binary digit를 Count 하라. 그 수는 음의 지수가 된다.
6. Characteristic 범위의 중간 지점(midpoint)에서 수(octal)는 곱하고, 그것을 수직선의 왼쪽으로 넣어라(8진수의 값은 200−2=176).

예 :

Characteristic	Fraction
01 111 110	100 110 001

$$= 2^{-2} \times \frac{1}{2} + \frac{1}{16} + \frac{1}{32} + \frac{1}{512}$$

$$= \frac{1}{4} \times [(256+32+16+1)/512]$$

$$= \frac{1}{4} \times 305/512$$

$$= 305/2048$$

$$= .1489 + (\text{or } .149)$$

Computer에서는 가산을 할 때 다음 예와 같이 Floating-point word를 취급한다.

Decimal Aritametic	Octal Aritametic	Floating-point Binary
12	14	00.1 100 또는
+97	+141	$2^4 \times 1100 + 00.1\ 100\ 001$
109_{10}	155_8	또는 $2^7 \times 1100001$

Computer에서 표준화된 Floating-point word의 취급은 :

Characteristic		Fraction
(Octal)	(Binary)	(Binary)
204	10 000 100	1100
207	10 000 111	1100001

Computer는 이들 두 Floating-point word를 하나의 공통적인 Characteristic을 가지게 하기 위해서 조정되고 또한 그들에 더해진다(add them).

$$2^7 \times (\frac{1}{2} + \frac{1}{4} + \frac{1}{16} + \frac{1}{32} + \frac{1}{128}) = 109$$

$$128 \times [(64+32+8+5+1)/128] = 109$$

간단한 가산의 예는 Instruction과 같이 Computer 내에서 Floating-point 실행에 적당하므로 감산(complement addition), 승산, 제산에서 발생하는 Fraction 혹은 Characteristic에 더 포함된 조작(manipulation)을 나타낼 수 없다.

Hexadecimal Floating-point Notation

2진법 Computer와 같은 System/360은 지수 0(zero)을 나타내기 위하여 Characteristic의 범위를 통한 Point midway를 사용한다. 지수는 Fraction(또한 hexadecimal에서 표현)이 원하는 값을 낳도록 증가시켜야 하므로 16의 승수이다.

System/360에서 조정된 것과 같이 10진수(149.25 처럼)를 Single floating-point word로 변환하기 때문에 다음과 같이 한다.

1. 수를 Integer와 Fraction으로 분리한다.
   분리 : 149.25=149 plus 0.25
2. 10진수 Integer를 16진수로 변환시킨다.
   즉 $149_{10} = 95_{16}$ (참고 Fig. 2-18)
3. 10진수 Fraction을 16진수로 변환시킨다.
   즉 $0.25_{10} = 0.4_{16}$ (참고 Fig. 2-19)
4. 둘을 결합하여 일반형으로 표현한다(fraction time 지수를 base 16처럼).
   $95.44_{16} = (0.954 \times 16^2)_{16}$
5. 64는 Characteristic 범위의 Midpoint이므로 지수(2)를 Characteristic을 얻도록 64에 더한다.
   64+2=66=1000010
6. Fraction을 2진수로 변환하고. 또한 그것을 16진수로 분류하면
   $.954_{10} = .1001\ 0101\ 0100$
7. 10진수 149.25(16진수 95.4)에 대한 Floating-point word는 다음과 같이 나타낸다.

Sign*	0
Characteristic	1100 0010
Fraction	1001 0101 0100 0000 0000 0000

*Zero(0)는 149.25의 값이 양수이므로 0(zero)이 사용된다.

〈다음 호에 계속〉