광주/전남지역대학, 학번 : 202334-153942, 이름 : 임준희, 컴퓨터과학개론

1) 폰 노이만의 구조란 수학자이자 물리학자인 존 폰 노이만이 제시한 컴퓨터 아키텍처의 일종을 뜻한다. 폰 노이만의 구조는 산술논리연산장치와 제어장치를 담당하여 프로그램 실행의 핵심 구성 요소인 중앙처리장치(CPU)와 데이터를 저장하고 처리하는 데이터 메모리(Data Memory), 모니터, 키보드, 마우스와 같은 사용자와 컴퓨터의 상호작용을 할 수 있는 입출력 장치(I/O)로 구성 되어있다.

폰 노이만 구조의 흐름을 설명하자면, 먼저 키보드, 마우스, 마이크 등과 같은 입력장치를 통해 외부의 프로그램과 데이터를 내부로 받아들인다. 그렇게 외부의 프로그램과 데이터가 동일한 메모리에 저장되면 프로그램과 데이터는 0과 1의 순차적인 명령어의 나열로 표현된다. 그렇게 CPU는 메모리에 저장된 순차적인 명령어의 집합을 읽고 해석하여 산술연산과 논리연산과 같은 동작을 통해 데이터를 처리하게 된다. 그렇게 처리한 데이터는 메모리에 다시 저장된다. 그 다음 모니터, 프린터기, 스피커 등과 같은 출력 장치를 통해 내부에서 처리된 데이터를 외부의 사용자와 상호작용 할 수 있게 되는 구조이다.

폰 노이만 구조는 프로그램과 데이터가 동일한 메모리에 저장 되는 것이 핵심으로 현대 대부분의 컴퓨터는 이와 같은 폰 노이만 구조의 시스템을 사용하고 있다. 폰 노이만 구조가 나오기 전에는 프로그램과 데이터가 다른 공간에 저장 되는 방식이였다. 폰 노이만 구조는 메모리에 저장된 명령어는 순차적으로 실행되어 한 번에 하나의 명령어를 읽고 해석하고 실행하게 된다. 이러한 점으로 인해 장점이 있는데 명령어가 하나씩 순차적으로 실행되기 때문에 상대적으로 간단하며, 또한 컴퓨터에 다양한 다른 작업을 할 경우에는 일일이 전선을 교체 해야 하는 하드웨어를 재배치하지 않고 프로그램만 교체하면 되는 장점이 있다. 하지만 단점도 있는데, 바로 폰 노이만 병목현상이다. 병목 현상이란 마치 교통 체증과 같이 어떤 특정 요소의 한계로 인해 전체가 제한을 받는 현상을 말하는데 이런 현상은 명령어가 하나씩 순차적으로 읽고 해석하고 실행되는 폰 노이만 구조에서도 나타나게 된다. CPU와 메모리간의 데이터 전송률의 한계로 인해 발생하는데 이러한 폰 노이만 병목 현상을 해결하기 위해 기존의 것에 변형을 한 하버드 아키텍처가 등장 했다. 병목현상이 있었던 폰 노이만의 구조와는 다른 방식으로 명령어를 저장하는 곳과 데이터를 저장하는 곳으로 메모리를 분리했다. 하버드 아키텍처는 병목현상을 좀 해소해 줄 뿐이고 결국엔 폰 노이만의 구조를 기반으로 한 것이기 때문에 프로그램을 순차적으로 처리하는 방식은 변하지 않는다. 따라서 현대의 컴퓨터는 외부는 폰 노이만 구조를 적용, CPU내부는 속도를 향상시키기 위해 하버드 아키텍처를 적용한 경우가 있으며, 고속 컴퓨팅을 요하는 프로세서는 데이터를 병렬 처리하고 프로그램을 실행하면서 데이터를 처리할 수 있는 구조인 하버드 아키텍처를 사용한다. 하지만 폰 노이만 구조도 여전히 개인용 컴퓨터와 같은 대부분의 컴퓨터에서 적용되고 있으며 구현이 간단하고 프로그래밍이 상대적으로 쉽기 때문에 현재까지도 많은 개발자들이 익숙하게 사용하고 있다.

2) 아래 사진은 파이썬을 이용한 10진수의 정수부분을 r진수로 변환하는 코드이다.

따라서 10진수의 정수부분을 r진수로 변환하는 과정은 아래와 같다.

1. 10진수 숫자를 r로 나눈다.

2. 최종적으로 몫이 0이 될 때까지 나누는 과정을 반복한다.

3. 이때, 나머지 값이 10보다 크게 나올 경우는 알파벳 문자를 사용하여 나타낸다. 예시로, 10진수 11은 16진수로 B로 나타낼 수 있다.

4. 몫이 0이 되면 나머지 값들을 역순으로 나열하여 최종 결과를 얻는다.

십진수의 정수67을 16진수로 변환하는 과정을 설명하면, 먼저 67을 16으로 소인수 분해한다. 몫이 0이 될 때까지 반복하면 나머지는 3, 4가 나온다. 이 나머지를 역순하면 43이므로 십진수 67을 16진수로 변환한 결과는 이 나오게 된다.

다음 십진수의 정수67을 8진수로 변환하는 과정을 설명하면, 마찬가지로 67을 8로 몫이 0이 나오기 전까지 소인수 분해하는 과정을 거친다. 그러면 나머지는 3, 0, 1의 값이 나오게 되며 이 나머지를 역순하면 103이다. 그러므로 십진수 67을 8진수로 변환한 결과는 이다.

마지막으로 십진수의 정수67을 2진수로 변환화는 과정은 똑같이 67을 2로 몫의 값이 0이 될 때까지 반복해준다. 그러면 1100001이라는 나머지가 나오게 되며 이 나머지 값들을 역순하면 1000011이 된다. 그러므로 십진수 67을 2진수로 변환한 결과는이다.

3) 아래 사진은 파이썬을 이용한 10진수의 소수 부분을 r진수로 변환하는 과정이다.

따라서 십진수의 소수부분을 r진수로 변환하는 과정은 다음과 같다.

1. 십진수의 소수 부분을 r진수로 곱한다.

2. 나온 값의 정수부분을 문자열에 넣고 나머지는 소수 부분으로 한다.

3. 정수부분이 10이상인 경우 알파벳 문자로 나타낸다.

예시로, 정수부분이 12일 경우는 16진법으로 하여 알파벳 C로 표현한다.

3. 이 과정을 소수 부분이 0이 나오거나, 특정부분이 반복될 때까지 계속 해준다.

십진수의 소수 부분 0.675를 16진수로 변환하는 과정을 설명하면, 먼저 십진수의 소수 0.675와 16진수를 곱하는 과정부터 해준다. 그러면 10.8의 값이 나오게 되며 이때 정수 10은 16진법으로 바꾸어 알파벳 A로 표현하여 문자열에 넣고 나머지 정수 부분을 뺀 소수 부분은 0.8로 이어서 16진수를 곱해준다. 그러면 12.8이 나오며 이때 12는 16진법으로 바꾸어 알파벳 C로 표현하여 문자열로 넣는다. 이후는 AC의 반복이므로 0.AC (16진수, 반복)의 결과가 나오게 된다.

다음으로 십진수의 소수 부분 0.675를 8진수로 변환하는 과정은 마찬가지로 십진수의 소수 부분 0.675와 r진수를 곱하는 과정이다. 8진수로 변환하는 것이 목표이므로 0.675\*8을 해준다. 그러면 결과는 5.4가 나오므로 정수 부분은 문자열에 추가 해주고 나머지 부분은 정수부분을 뺀 0.4로 갱신하여 계속 계산해준다. 계속 갱신하여 계산하다 보면 0.54146 이후로는 계속 반복이 되므로 0.53146(8진수, 반복)의 값으로 마무리한다.

그리고 다음 십진수의 소수 부분을 0.675를 2진수로 변환하는 과정도 십진수의 소수 부분 0.675와 2진수(r진수)를 곱하는 과정부터 진행한다. 그러면 1.35 여기서 정수 부분은 문자열에 추가하고 나머지 소수부분은 정수부분을 뺀 0.35로 갱신하여 2진수를 곱해준다. 이 과정을 하다 보면 이후로는 10101이 반복되므로 0.10101(2진수, 반복)의 결과가 된다.