프로그래밍 언어 개론

1.1 컴퓨터란 무엇인가?

1.2 프로그래밍 언어와 언어 처리기

1.3 프로그래밍 언어의 변천

1.4 프로그래밍 절차

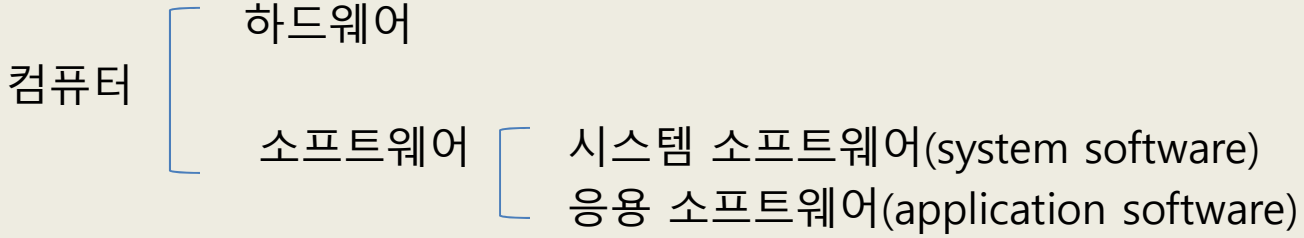
1.1 컴퓨터란 무엇인가?

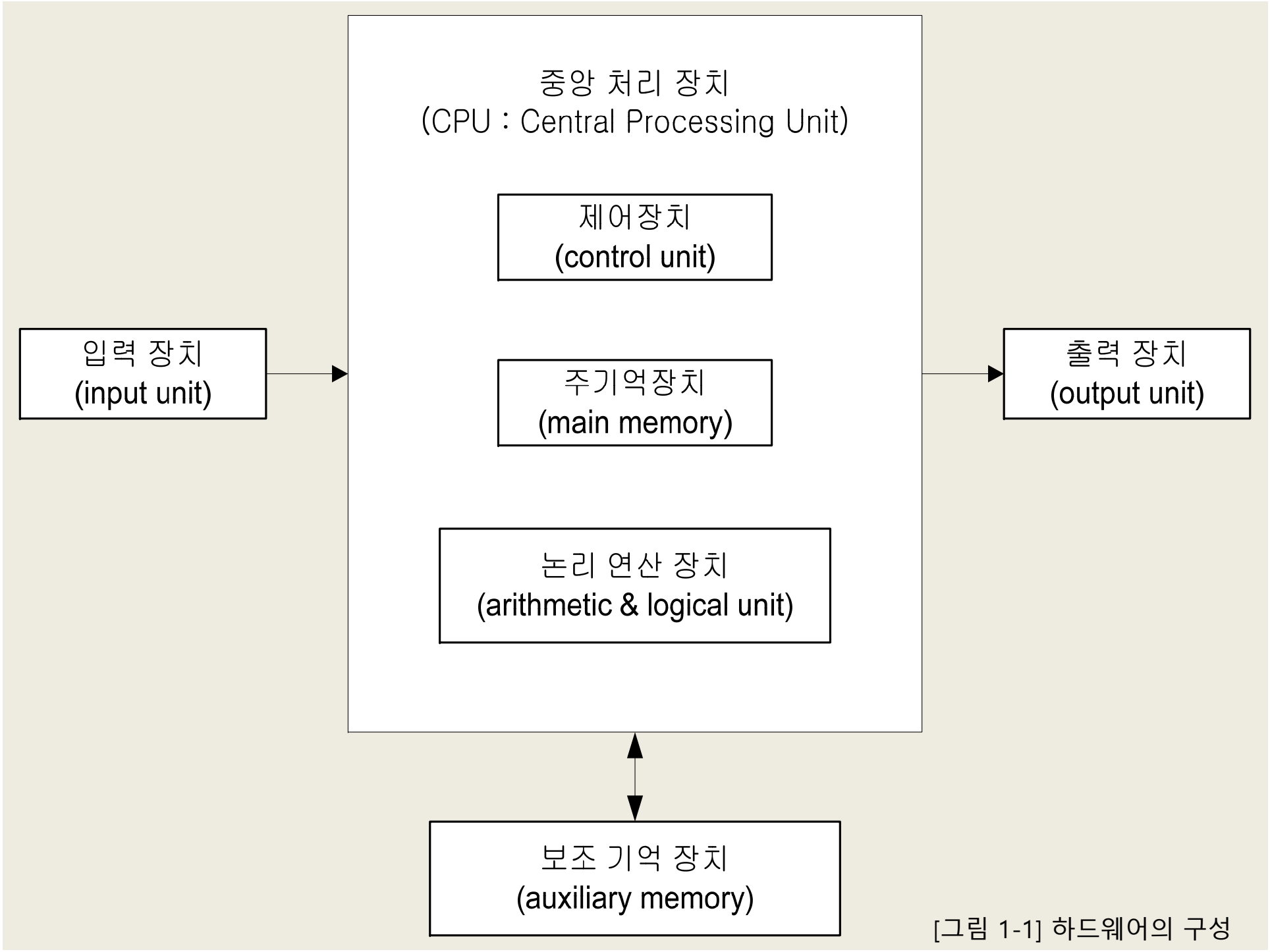
- ENIAC(Electronic Numerical Integrator And Computer)
  - 1940년대 중반 개발된 최초의 컴퓨터

- EDPS(Electronic Data Processing System) : 전자 자료 처리
  - 신속성, 정확성 , 대량의 데이터 처리
  - 처리 결과의 신뢰성을 향상
  - 다양한 업무의 종합적인 처리가 가능

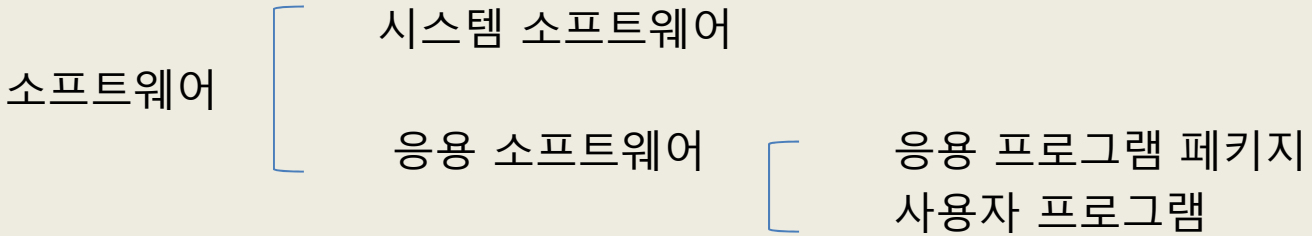
- "0"과 "1"의 두 값을 갖는 이진 체제(binary system)에 기본을 둔 시스템

- 하드웨어(hardware)와 소프트웨어(software)로 구성





[그림 1-1] 하드웨어의 구성



- 시스템 소프트웨어(system software)
  - 하드웨어를 제어하고 관리하기 위한 여러 가지 종류의 프로그램들을 총칭
  - 제어 프로그램(control program)과 처리 프로그램(process program)으로 구분

- 응용 소프트웨어(application software)
  - 응용 프로그램 패케지(application program package)와 사용자 프로그램으로 구분
  - 응용 패케지 : 비절차적 언어(non-procedural language)를 사용
    - 스프래드쉬트(spreadsheet), 워드프로세서(word processor),
    - 데이터 통신 소프트웨어, 통계처리 패키지,
    - 데이터 베이스 시스템(data base system)에 기초한 툴
  - 사용자 프로그램

1.2 프로그래밍 언어와 언어 처리기

1.2.1 프로그래밍 언어

- 사람과 기계 사이에 대화를 가능하게 해 주는 것
- 알고리즘이나 자료 서술을 위한 임의의 기호 약속을 의미
- 인간이 컴퓨터를 이용하여 수행하고자 하는 일을 컴퓨터에게 전달하기 위한 표기법.

(1) 기계어(machine language)
- '0' 과 '1'의 숫자로 구성
- 기계 중심적인 언어(machine oriented language)

(2) 어셈블리어(assembly language)
- 기계어에 해당되는 명령을 기호를 이용해서 나타낸 기호 언어
- 어셈블리어는 기본적으로 기계어와 일대 일로 대응되며,
- 기호를 이용하여 명령을 나타낸 관계로 인하여 상징어(symbolic language)라고도 함
- 기계 중심적인 언어로서 프로그램 시에 하드웨어에 대한 지식이 요구됨

(3) 고급언어(high-level language)
- 인간이 이해하기에 용이한 명령문이나 기호를 사용하여 구성된 프로그래밍 언어
- 문제 중심적인 언어(problem oriented language)

1.2.2 언어 처리기

- 프로그램이 기계에서 실행될 수 있도록 만들어 주는 시스템 프로그램
- 프로그래밍 언어로 작성된 프로그램을 기계어로 번역시켜 주는 프로그램을 번역기(translator)라고 함
- 번역기에 입력되는 언어를 원시 언어(source language), 입력되는 프로그램을 원시 프로그램이라고 함
- 출력되는 언어를 목적 언어(object language), 출력되는 동등한 프로그램을 목적 프로그램이라고 함

(1) 컴파일러(compiler)

- 원시 언어가 고급 언어이고 목적 언어가 기계어인 번역기를 컴파일러라고 함
- 고급언어를 기계어로 번역해 주는 프로그램
- 컴파일러가 프로그램을 번역하는 과정을 컴파일(compile)이라고 함

(2) 어셈블러(assembler)

- 원시 언어가 상징어 혹은 어셈블리어이고 목적 언어가 기계어인 번역기
- 어셈블리어로 작성된 프로그램을 기계어로 번역해 주는 프로그램

(3) 인터프리터(interpreter)

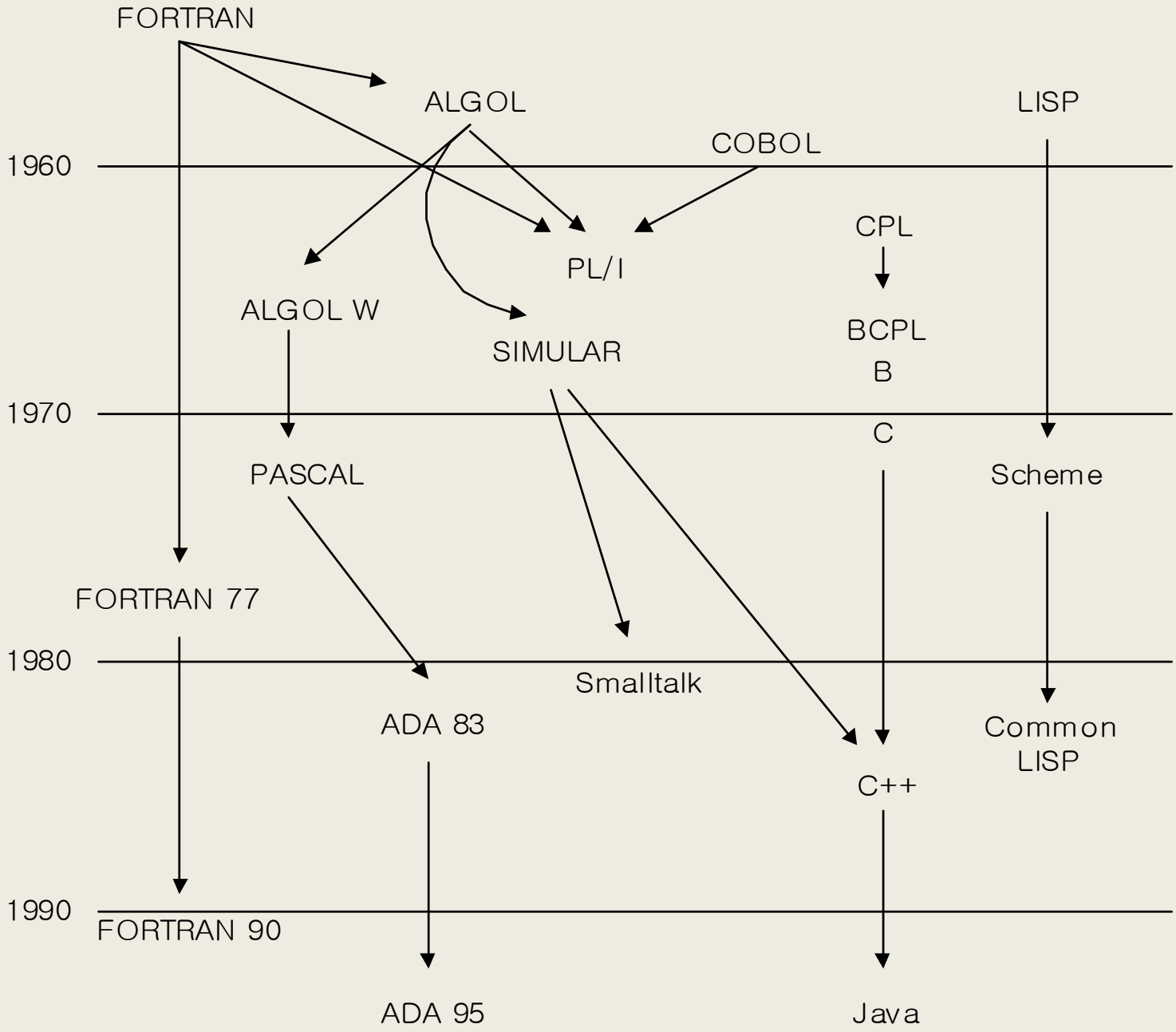
- 원시 프로그램을 한번에 모두 기계어로 번역하는 것이 아니고 매 명령문마다 매번 해석하여 처리해 주는 언어 처리기
- 대화식(interactive) 작업에 적합한 언어 처리기임

(4) 하이브리드 기법

- 컴파일러와 인터프리터의 혼합 개념으로 프로그램을 실행하는 방법
- 프로그래밍 언어를 순수 기계어가 아닌 적당한 중간단계 언어로 번역한 후에 소프트웨어로 인터프리팅하는 방법
- Java와 같은 언어가 이와 같은 하이브리드 기법을 택한 대표적인 언어의 예

1.3 프로그래밍 언어의 변천

- 프로그래밍 언어는 1950년대 FORTRAN, COBOL, ALGOL, LISP을 필두로 해서 수많은 언어가 개발되어 사용됨
- 초창기 언어들은 특정분야의 처리를 주목적으로 개발됨
- 초창기 언어들은 이후 다른 언어들의 개발에 영향 줌
- [그림 1-2]는 유명했던 프로그래밍 언어들의 계통도를 나타냄
  - 그림에서 화살표는 프로그래밍 언어간에 영향을 주고받은 관계를 의미



[그림 1-2] 프로그래밍 언어 계통도

(1) 1950년대 : 초기의 프로그래밍 언어

- 현재 고급언어의 기본이 되는 대표적인 초창기 언어들이 개발된 시기
- 특정 영역의 문제를 해결하기 위한 언어로 개발된 특성을 갖음
- 다음 세대에 만들어진 언어들의 기본개념이 됨

① FORTRAN(FORmula TRANslation)

- 1957년 Backus 등에 의해 IBM에서 개발된 언어로서 과학 기술용 수치 계산 언어
- 문법이 간단하고 내장함수가 많아서 수학적인 계산을 위한 프로그램 작성에 많이 이용

② COBOL(COmmon Business Oriented Language)

- 1960년 미국 CODASYL에서 개발된 언어로서 상업적 사무 처리용 언어
- 자연어의 형태와 유사한 구문을 사용하는 특징을 가지며 일반 자료 처리에 적합한 언어

③ ALGOL(ALGOrithmic Language)

- 1960년 IFIP에서 개발된 언어로서 알고리즘을 기술하기 위한 언어
- 알고리즘을 효율적으로 기술하기가 용이하다는 장점을 가지며 이후에 개발된 언어의 기초를 제공

④ LISP(LISt Processor)

- 1950년대 후반 MIT에서 만들어진 언어
- LISP은 초창기 인공지능 문제를 해결하기 위한 언어로서 널리 사용

(2) 1960년대 : 언어의 범람

- 이 시기는 많은 언어가 만들어지고 사라져간 시기
- 특별한 관심분야에 맞추어져 언어들이 양산됨
- 모든 언어의 개념을 갖는 마지막 언어를 만들고자 하는 노력들이 진행됨

① PL / I (Programming Language One)

- 1964년 IBM에서 개발한 언어
- FORTRAN, COBOL, ALGOL 등의 기존 언어의 장점만을 따서 만듬
- 병행성, 예외처리, 기억장소할당 등 새로운 개념을 도입한 언어

② Simular

- 노르웨이의 노르웨이 컴퓨터 센터에서 시뮬레이션을 목적으로 개발된 언어
- 객체지향 언어의 기본이 되는 class의 개념을 처음으로 사용
- 이후에 개발되는 객체지향 언어들은 Simular로부터 파생되어 짐

(3) 1970년대 : 개념이 명확한 언어들의 개발

- 프로그램의 간결성, 추상화 등의 개념이 포함된 언어들이 대두된 시기
- 대표적인 언어로서 PASCAL, C 등의 출현

① PASCAL

- 1971년 Wirth에 의해 개발된 언어
- 기존의 Algol과 같은 언어보다 작고 간결하며 효율적으로 구성한 구조적 프로그래밍 언어

② C

- 1972년 Dennis Ritchie에 의해 개발된 언어로서 시스템 소프트웨어 개발을 목적으로 만들어진 언어
- C는 저급 언어와 고급 언어의 장점을 취해 만든 언어
- 일반 상용 시스템의 개발을 비롯한 많은 분야에서 폭 넓게 사용되는 언어

(4) 1980년대 : 새로운 방향의 시도

- 1980년대의 가장 큰 이정표는 미국 국방성에서 개발한 Ada를 들 수 있다.
- 기존의 언어 개념에서 벗어난 언어들이 개발되고 연구된 시기
- 함수언어(functional language), 논리언어(logical language), 객체지향언어(object oriented language) 등의 개념을 갖는 언어들의 발전이 이루어 짐

(5) 1990년대 : 웹 언어의 대두

- 1990년대는 1980년대부터 대두되었던 간결한 언어, 데이터베이스를 쉽게 처리할 수 있는 언어, 간단한 프로그램을 사용자가 직접 처리할 수 있도록 하기 위한 언어 등을 추구하는 4세대 언어(4 GL) 개념의 대두

- 인터넷의 보급으로 인해 웹과 관련된 언어의 개발이 하나의 과제로 대두된 시기

- 대표적인 언어의 형태로 Java의 탄생

- Java는 Sun 마이크로시스템즈에서 개발된 언어로서 처음에는 가전제품을 위한 소프트웨어 개발을 목적으로 개발되었으나 1993년부터 World Wide Web이 폭 넓게 확산되면서 Web은 Java의 가장 일반적인 응용분야로 자리 잡게 됨

- 1990년대 중반 이후 웹 서버 프로그램을 효율적으로 개발하기 위한 서버 측 스크립트 언어(server-side script language)들이 경쟁적으로 개발 됨
  - 서버의 종류에 구애받지 않는 오픈된 소프트웨어인 PHP(Professional Hypertext Preprocessor),
  - Microsoft에서 개발한 ASP(Active Server Page),
  - Sun Microsystems에서 개발한 JSP(Java Server Page)

1.4 프로그래밍 절차

(1) 문제분석

- 처리하고자 하는 업무를 분석하여 컴퓨터를 이용한 처리의 타당성, 가능성, 능률성 등을 검토하고, 문제와 처리의 범위를 정한다.

(2) 입출력 설계

- 주어진 문제에 의해 입력으로 주어질 수 있는 자료를 결정하고 출력하고자 하는 항목을 결정한다. 또한, 문제 해결에 필요한 자료구조 등을 정의한다.

(3) 순서도 작성

- 문제를 해결하기 위한 논리적인 절차와 흐름의 방향을 기호를 이용하여 나타낸다. 이를 순서도(flowchart)라고 하는데, 순서도를 이용하여 전체 프로그램의 논리를 단계적으로 표현함으로써 작업 전체의 흐름을 손쉽게 파악할 수 있다.

(4) 코딩과 프로그램 입력

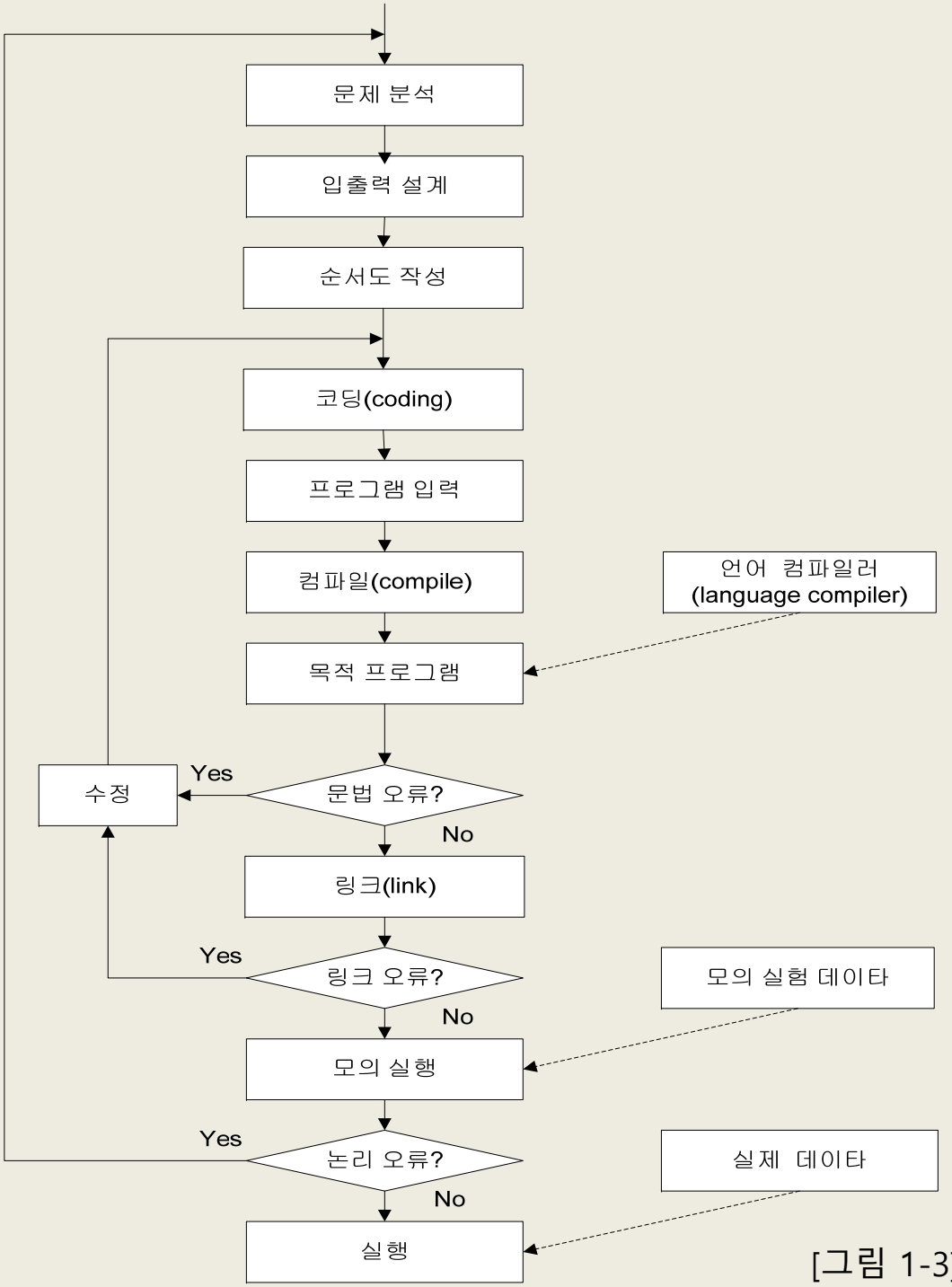
- 순서도로 나타내진 문제 해결 방법을 주어진 프로그래밍 언러로 나타내는 것을 코딩(coding)이라고 한다. 주어진 순서도에 대한 코딩 작업이 완료되면 프로그램을 에디터(editor) 등을 이용하여 기계에 입력

(5) 컴파일 및 링크

- 입력된 프로그램을 기계가 인식하기 위해서는 기계어 수준의 언어로 번역해 주어야 한다. 이때 사용되는 시스템 소프트웨어가 컴파일러(compiler)와 링커(linker)이다.
- 컴파일 과정에서 문법오류(syntax error)가 발생되면 작성된 프로그램의 문법이 틀린 것이므로 프로그램을 수정하여 다시 컴파일을 하여야 함

(6) 모의 실행 및 실행

- 모든 과정이 완료되면 모의 실험 데이터를 이용하여 작성된 프로그램을 실행하게 되는데 이때 출력된 결과를 가지고 프로그램의 논리적 타당성을 검토
- 모의실험 데이터는 프로그램에서 발생될 수 있는 모든 경우의 수를 포함한 셈플 데이터로 구성
- 이와 같은 작업 과정에서 원하는 결과가 출력되지 않는다면 이는 프로그램이 논리적 모순에 빠졌음을 의미함
- 이러한 오류를 논리 오류(logical error)라고 하며, 이와 같은 경우 문제 분석 단계부터 자세한 검토가 다시 이루어져야 함
- 이와 같은 과정을 거쳐서 모의 실험 결과가 올바르게 나타나면 실제 데이터를 가지고 전체적인 결과를 얻는다.



[그림 1-3] 프로그래밍 절차