MiniC 컴파일러 만들기(5): Code Generation: x86

컴파일러, 11번째 시간

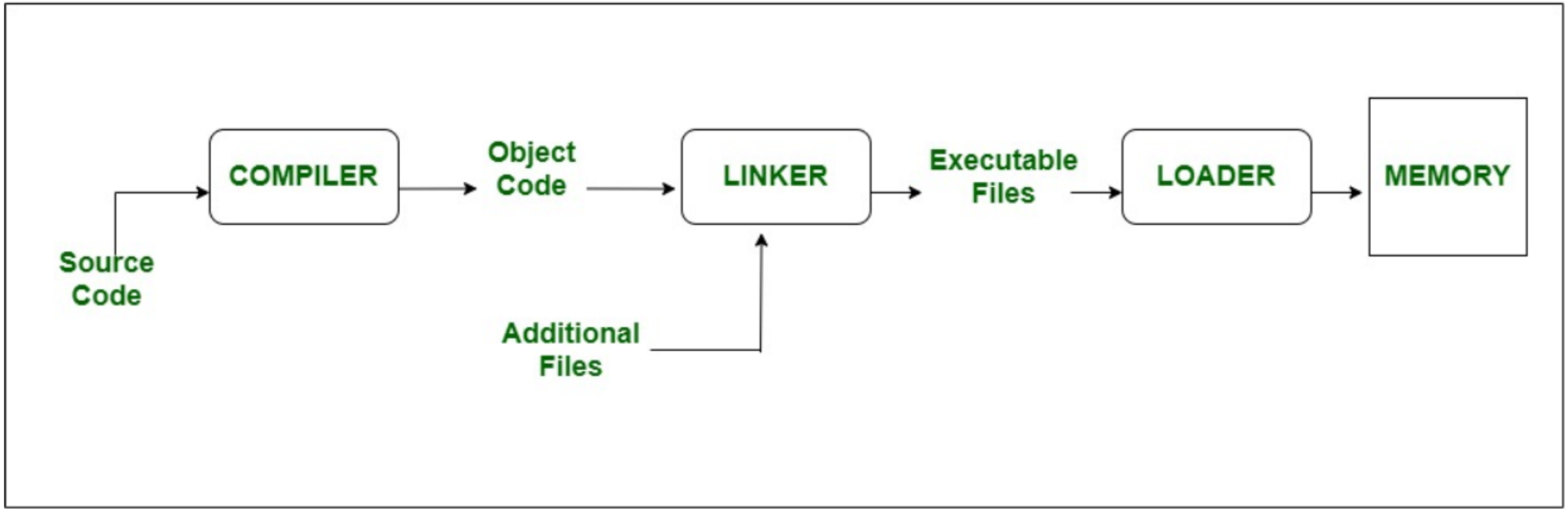
경민기, 2025-11-25

오늘 수업 내용

- 1단계 MiniC 문장을 x86-64 어셈블리로 변환하는 과정을 이해한다.
- 스택 프레임, 로컬 변수, 함수 호출 규약을 이해한다.
- if, while, for 제어문을 레이블과 분기 명령으로 바꾸는 방법을 본다.
- printf / scanf 호출로 C 라이브러리와 연동하는 방법을 살펴본다.

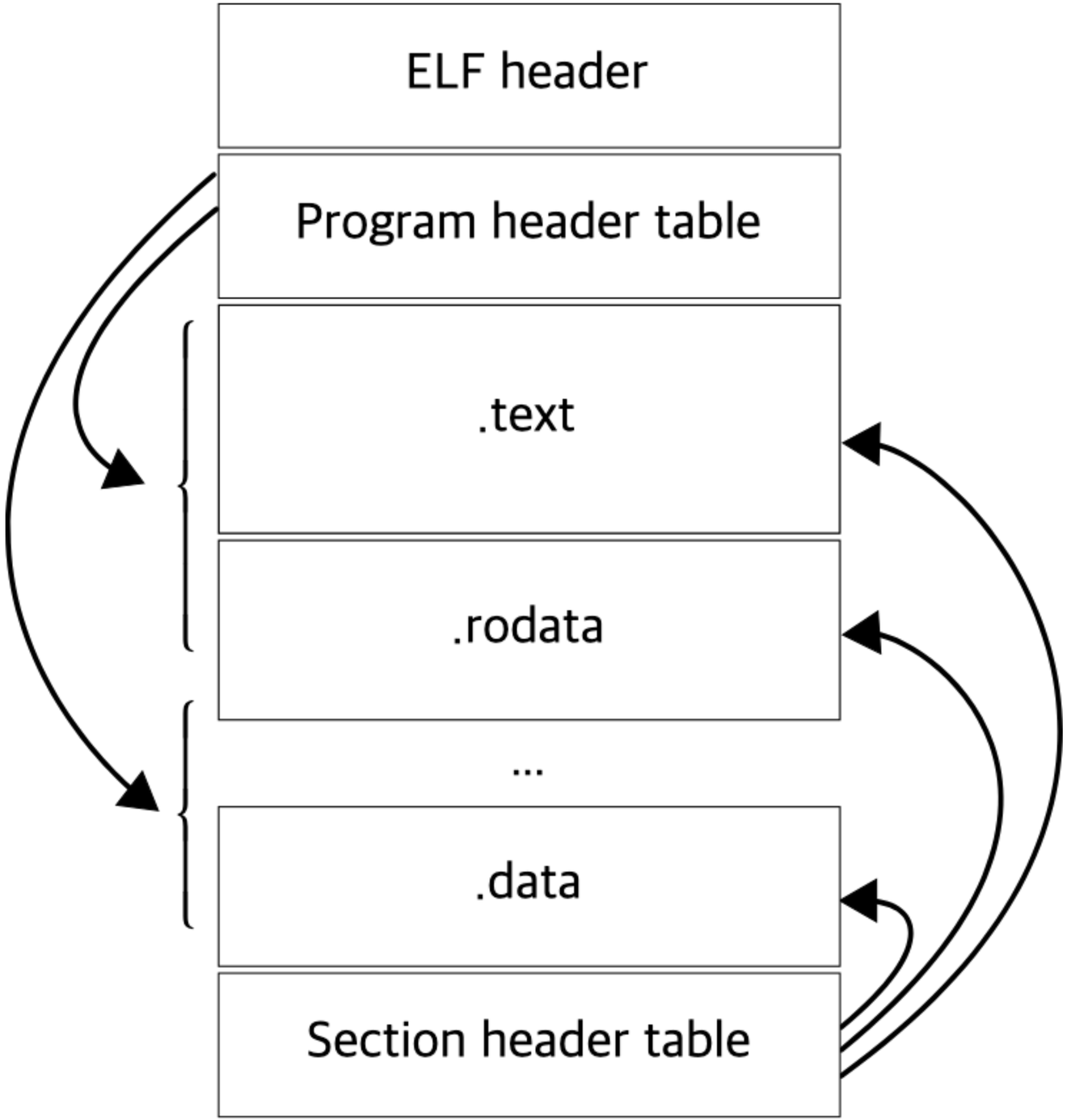
ELF 파일 포멧

프로그램 로딩 순서


COMPILER
Object Code
LINKER
Executable Files
LOADER
MEMORY
Source Code
Additional Files

ELF

- ELF(Executable and Linkable Format)는
  Linux/Unix 실행파일의 표준 바이너리 포맷
- 목적 파일(.o), 공유 라이브러리(.so), 실행 파일(a.out / ELF)이 모두 동일 형식 기반
- 링커와 로더가 협력하기 위한 구조적 설계



x86 중간코드

레지스터

- 64비트 모드(x86-64)에서는 “일반 목적 레지스터”가 여러 개 있음:
  - rax, rbx, rcx, rdx, rsi, rdi, rbp, rsp
  - r8 ~ r15
  - 각각은 64비트(8바이트) 레지스터고, 하위 32비트(eax), 16비트(ax), 8비트(al)처럼 잘라 쓸 수도 있음

- 우리가 쓰는 리눅스 x86-64(SysV ABI)에서 관습적으로 다음을 사용함:
  - 함수 반환값 → rax에 넣어서 리턴
  - 스택 포인터 → rsp
  - 프레임 포인터 → rbp

- 일부 레지스터는 “함수 호출해도 값이 유지돼야 하는 레지스터(callee-saved)”,
- 나머지는 “마음대로 써도 되는 레지스터(caller-saved)”로 약속되어 있음.

x86코드 생성 시의 규칙

- 표현식(expr)의 결과는 항상 eax 레지스터에 둔다.
- 문장(stmt)은 변수 선언, 대입, 제어문, 함수 호출로 본다.
- 제어문은 레이블(.Lbegin, .Lend)과 분기(jmp, je)로 표현한다.
- printf / scanf는 C 라이브러리 함수 호출로 처리한다.