Подзадача 1: извлечение отладочной информации формата DWARF о типах аргументах функции и о возвращаемом значении

Реализованы несколько функций, позволяющих получить информацию об аргументах и типе возвращаемого значения

1. get_ret_type_by_fname() – используется для извлечения информации о возвращаемом значении:
    - Изначально мы находимся в секции .debug_info
    - Выбирается смещение от .debug_info, чтобы прийти в compilation_unit, в котором находится искомая функция
    - Выполняется полный обход дерева DIE (аналогично `naive_address_by_fname`)
    - Запись должна иметь тэг DW_TAG_subprogram
    - Извлекается атрибут DW_AT_type, который описывает возвращаемое значение функции (при наличии для каждой записи-подпрограммы)
    - В случае, если есть совпадение имён (запись описывает именно искомую функцию), то есть используется атрибут DW_AT_name, то найденное значение выводится с помощью print_type (С помощью DW_AT_name + form мы извлекаем запись, которая соответствует типу и далее выводим его с помощью рекурсивной функции print_type)
2. get_arguments_by_fname() – используется для извлечения информации о типах аргументов
    - Выполняется полный обход дерева DIE
    - Запись должна иметь тэг DW_TAG_subprogram
    - Продолжаем, только при совпадении имён
    - Извлекаем дочерние записи, содержащие тэг DW_TAG_formal_parameter
3. print_type() – используется для вывода информации о типе
    - Рекурсивная функция, которая обходит записи описывающие типы.
    - Выводятся модификаторы типа: например, указатель, константа, typedef
    - Условие остановки – базовый тип или user-defined тип (структура)

Декларация функции	**Вывод**
`void *` `memset(void *v, int c, size_t n)`	-> (pointer void) (pointer void , int , typedef long unsigned int)
`void` `monitor(struct Trapframe *tf)`	-> (void) (pointer struct Trapframe)
`uint16_t` `cmos_read16(uint8_t reg)`	-> (typedef short unsigned int) (typedef unsigned char)
`pde_t *` `alloc_pd_early_boot(void)`	-> (pointer typedef typedef long unsigned int) ()

Подзадача 2: вызов библиотечных функций по имени

Реализована возможность вызова функций из монитора с любым количеством аргументов, в том числе и функции с переменным числом аргументов. Поддержка строк, char, int…

Реализация вызова функции по имени с помощью ассемблерной вставки

1. С помощью bind_function, вычисляется адрес функции
2. С помощью ассемблерной вставки задаём правильные значения регистров и формируем стек согласно соглашению о вызовах, используемых в JOS. Оно такое же как в Linux:
    - Первые целые 6 аргументов функции идут на регистры %rdi, %rsi, %rdx, %rcx, %r8, %r9, остальные помещаются на стек
    - Первые 7 SSE аргументов (float и double) помещаются на на регистры %XMM0 - %XMM7, остальные помещаются на стек (в JOS не работает)
3. В случае, если нужно, извлекается значение %rax регистра – возвращаемое значение функции

Ввод:	**Вывод:**
call test cputchar 1 c x	x
call test strcmp 2 s text s text	Result: 0
call test monitor 0	Запуск монитора
call test cprintf 12 s Supported args: %c, %d, %ld, %lld, %u, %lu, %llu >%s< %d %d %d! c x d 42 ld 12345678900 lld 12345678900 u 4000000000 lu 60000000000000 llu 6000000000000000 s	Supported args: x, 42, 1234567890, 123456787890, 4000000000, 600000000000, 6000000000000000 >another text< -1 0 147! Out: 116

another text d -1 d 0 d 147	