class Bird {
    private:
        int age;
        double weight;
        char favorite_letter;
    public:
        void eat();  // In C++ the programmer specifies
        virtual int sleep();  // "virtual" when dynamic binding is
        virtual void speak();  // desired. (Otherwise static binding is
        // used by default.)
}

class Eagle : public Bird {
    private:
        int zip_code;
    public:
        virtual void speak();  // overrides Bird version
        virtual double findfish();
        void look_important();
        virtual void buildnest();
}

foo() {
    Bird B;  // In C++ this allocates a Bird object on the stack
    Eagle E;  // In C++ this allocates a Eagle object on the stack
    Bird *b_ptr;
    b_ptr = &E;

    b_ptr->speak();
    b_ptr->sleep();
    b_ptr->eat();
}

Draw a picture of B and E, including how data members would be laid out and any mechanisms that support dynamic method binding (dynamic dispatch). Show pseudo-assembly code for the function calls on b_ptr.
b_ptr->speak();

v_table_addr = &b_ptr
fun_addr = *(v_table_addr + offset)
call fun_addr