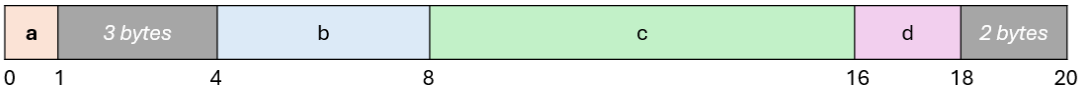
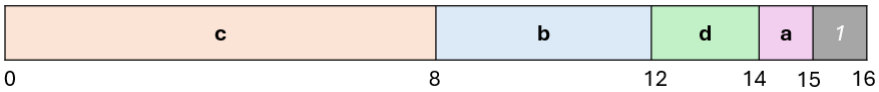
Câu	Đáp án	Giải thích
1	B	Dùng biểu diễn bù 2 của x (~x + 1) sẽ có biểu diễn của -x (với mọi x nguyên)
2	A	Giá trị trả về của hàm trong kiến trúc 64bit được lưu trong thanh ghi %rax
3	D	Trong gọi hàm trong kiến trúc 64 bit, 6 tham số đầu tiên được truyền qua thanh ghi, từ tham số thứ 7 được đưa vào stack. Với hệ thống 32bit, tất cả tham số được đưa vào stack.
4	B	Với lệnh pushl %ebp trong 32 bit, %esp bị giảm xuống 4 đơn vị, %ebp (thanh ghi source của lệnh push) không đổi
5	C	Stackframe hàm mẹ nằm ở địa chỉ cao hơn stackframe hàm con, được cấp phát trước và thu hồi sau stackframe hàm con. Các hàm không thể truy xuất biến cục bộ trong stackframe của hàm khác.
6	A	Với mảng short a[5], ta có &a[i] = &a + i*2 = %eax + %edx*2 nên ta có biểu thức địa chỉ là (%eax, %edx, 2). Lưu ý a[i] có kích thước 2 byte, cần dùng movw để lấy đúng 2 byte tại ô nhớ của a[i], không lấy dư.
7	B	%ebp trỏ đến vị trí lưu old %ebp của hàm mẹ, ô nhớ lưu địa chỉ trả về là vị trí %ebp + 4, tham số thứ nhất lưu ở %ebp + 8. Trong stack không lưu giá trị trả về của hàm, giá trị này lưu trong thanh ghi.
8	C	Kích thước của mảng 2 chiều: sizeof(T)*R*C = 2*5*5 = 50 bytes
9	C	Sau lệnh 1, %esp giảm 4 còn 0x1024. Sau lệnh 2, %esp giảm 20 (= 0x14) nên kết quả là 0x1010.
10	B	Yêu cầu căn chỉnh chung K của struct là yêu cầu căn chỉnh lớn nhất của các thành phần trong struct. float a có kích thước kiểu dữ liệu là 4, K(a) = 4. double b có kích thước 8, trong Linux 32 bit (trường hợp ngoại lệ) thì K(b) = 4. char c có kích thước 1, K(c) = 1. Vậy yêu cầu căn chỉnh chung K của struct là max{4,4,1} = 4
11	C	(Lý thuyết) Khi cấp phát struct, các thành phần được cấp phát theo đúng thứ tự khai báo trong struct. Thành phần nào khai báo trước được cấp phát trước và nằm ở các ô nhớ có địa chỉ thấp hơn, các thành phần càng phía sau (cấp phát sau) thì địa chỉ lưu càng cao.
12	B	Vẽ cấp phát struct (có alignment) như hình dưới. Lưu ý khi có alignment, cần đảm bảo tổng kích thước struct chia hết cho K chung của nó (= 4), nếu chưa thì thêm byte trống (màu xám).  Vậy kích thước là 16 bytes.
13	A	Khi không căn chỉnh, tổng kích thước struct bằng tổng kích thước các thành phần cộng lại: 4 (float) + 8 (double) + 1 (char) = 13 bytes. Vậy chênh lệnh 3 bytes so với trường hợp có căn chỉnh.
14	B	Kích thước union là kích thước của thành phần có kích thước lớn nhất. float a có kích thước 4 bytes, char c có kích thước 1 bytes, double b có kích thước 8 bytes. Vậy kích thước union là 8 bytes.
15	D	Các thành phần union đều có địa chỉ bắt đầu là địa chỉ của union. Như vậy địa chỉ của thành phần c là (%eax). Thành phần c là kiểu char (1 byte) nên để lấy đúng 1 byte cần dùng movb. Câu lệnh đúng: movb (%eax), %bl (ĐÁP ÁN KHÁC)
16	A	Để phòng tránh: dùng fgets với giới hạn phù hợp, dùng canary (mặc định được bật hoặc dùng -fstack-protector). -fno-stack-protector là không sử dụng canary để bảo vệ file.

17	B	Ô nhớ trỏ đến bởi %ebp là ô nhớ lưu old %ebp của hàm mẹ, ghi đè tùy ý sẽ lỗi. Ngoài ra ghi đề ô nhớ chứa địa chỉ trả về cũng gây lỗi. Mặt khác, chuỗi input chỉ ghi đè được các ô nhớ nằm ở địa chỉ CAO hơn nó.
18	C	Các hàm và biến toàn cục, dù có static hay không đều là symbol. Với biến cục bộ trong hàm, nếu có static thì là symbol, biến cục bộ thường (không static) không phải symbol.
19	A	Biến cục bộ (không static) là biến cục bộ thường, được lưu trong stackframe của các hàm.
20	D	Linker không thể liên kết 2 file source có định nghĩa 2 hàm trùng tên do đây là trường hợp phân giải symbol trùng tên, 2 hàm đều là 2 strong symbol nên lỗi linker. Mặt khác linker nhận đầu vào là file .o, ngay cả chỉ có 1 file .o vẫn phải dùng linker (như Lab 2 và 3). Linker mặc định làm việc với hàm và biến toàn cục.
21	C	%eax = 0x120. %ah là thanh ghi 1 byte nhỏ nằm trong %eax, chứa byte thứ 2 từ phải đếm sang, tức là 0x1, %ah = 0x0 thì %eax = 0x20. Lệnh 3 tăng %eax lên 1 thì %eax = 0x21.
22	B	Biến cục bộ **có static** sẽ lưu trong section .bss (nếu chưa gán giá trị) hoặc .data (nếu đã gán giá trị). Do đã gán a = 5 nên nằm trong .data. Section .symtab chỉ chứa tên symbol chứ không lưu biến.
23	D	Mảng có kích thước khác nhau trong 32 bit và 64 bit là do kích thước kiểu dữ liệu khác nhau giữa 2 hệ thống. long hoặc kiểu pointer là kiểu dữ liệu như vậy.
24	C	Trường hợp bình thường: tổng kích thước 20 bytes (K chung = 4)  Trường hợp tối ưu: sắp xếp lại các thành phần trong struct theo thứ tự giảm dần của kích thước dữ liệu, ta có struct ex {double c; char* b; short d; char a;} có tổng kích thước 16 bytes (K chung = 4)  Như vậy giảm được 4 bytes.
25	C	sym1 có trong symbol table, tức là nó là 1 symbol. Cột Bind cho biết kiểu symbol là GLOBAL hay LOCAL. Mặt khác Ndx cho biết vị trí định nghĩa symbol trong file .o, nếu là UND (Undefined) nghĩa là không có định nghĩa của symbol trong file .o hiện tại, mà nằm ở file khác (External symbol).