Bài 3: Kiến trúc bộ lệnh

Câu 1:

a)

```
addi    $s1, $zero, 5
addi    $s2, $s1, -2
sll     $s3, $s1, 2
sll     $s4, $s2, 1
add     $t0, $s1, $s3
add     $t1, $s2, $s4
sub     $s5, $st0, $t1
```

b)

```
lw    $t0, 20($s3)
add    $s1, $s2, $t0
```

Câu 2:

*Khác nhau: Trong Big End, byte cao nhất được lưu trữ đầu tiên, còn đối với Little End, byte thấp nhất được lưu trữ đầu tiên.
*Đối với MIPS, có thể dùng cả 2, nhưng theo mặc định MIPS dùng Little End.

Câu 3:

x	y	x + y
4 = 00000100 12 = 00001100 19 = 00010011 60 = 00111100	-6 = 11111010 -9 = 11110111 -12 = 11110111 88 = 01011000	4 + (-6) = 11111110 = -2 12 + (-9) = 00000011 = 3 19 + (-12) = 00000111 = 7 60 + 88 = 10010111 = 148

Câu 4:

- Các định dạng lệnh MIPS: op, rs, rt, rd, shampt, funct.
- Biểu diễn: add &t0, $s1, $s2

op	rs	rt	rd	shampt	funct
000 000	10001	10010	01000	00000	100000

Câu 5: Các bước chuyển 1 chương trình C thành mã máy:
B1: Biên dịch chương trình C thành mã ngôn ngữ máy (assembly)
B2: Dịch mã assembly thành mã máy (machine code) bằng trình dịch assembler.
B3: Liên kết các tệp đối tượng được tạo ra từ mã nguồn với các thư viện cần thiết để tạo thành chương trình hoàn chỉnh.
B4: Đưa chương trình ngôn ngữ máy vào bộ nhớ máy tính.

Câu 6:

```
slti $t0, $s0, 0
beq $t0, $zero, else
addi $s1, $zero, 0
j exit
```

```
else:
    add $s1, $zero, $s0
Exit
```

Câu 7: Các bước thực thi thủ tục:

B1: Đặt các tham số ở một nơi mà thủ tục có thể truy xuất được.

B2: Chuyển quyền điều khiển cho thủ tục

B3: Yêu cầu tài nguyên lưu trữ cần thiết cho thủ tục đó.

B4: Thực hiện công việc (task).

B5: Lưu kết quả ở một nơi mà chương trình có thể truy xuất được.

B6: Trả điều khiển về vị trí mà thủ tục được gọi. Vì một thủ tục có thể được gọi từ nhiều vị trí trong một chương trình.

Câu 8: Cấu trúc thủ tục:

Phần đầu:

- Khai báo kích thước cho stack.
- Lưu các thanh ghi cần thiết.

Phần thân:

- Thực hiện chức năng yêu cầu (có thể gọi các thủ tục khác).

Phần cuối:

- Phục hồi các thanh ghi cần thiết.
- Xóa stack.