

1. (a) 32-bits
 (b) 2^{32} bytes = 4GB
 (c) 8, 32-bits
 (d) 32-bits, marca la posición de la siguiente instrucción
 (e) Porque tiene qe poder "llevarte" a cualquier lugar en la memoria
2. (a) 32-bits, y guarda los status flags (resultado de operaciones aritméticas), un flag de control ("endianess" de los strings), y flags de sistema (activan y desactivan cosas del OS)
 (b)

11, Prendido si el número es de un valor muy grande para el destino

Overflow: 7, Vale lo mismo qe el bit más significativo del resultado

Interrupt: 9, Marca si el procesador va a responder a interrupciones

(c) No es lo mismo, los primeros 32bits están reservados. Y cambia de nombre

3. (a) Se usa para pasar parámetros entre subrutinas, y para guardar valores dentro de mi subrutina
 (b) El procesador lo ubica en la memoria (en base al necesitado). Y dentro de mi memoria asignada, arriba de todo
 (c) Apunta al último elemento +1 de mi stack frame
 (d) Apunta a una posición dentro del stack, se suele usar para marcar la base del stack frame
 ebp: 6
 esp: 4

13:

12:

11:

10:+EIP

9:+EBP j- MI EBP

8:+ j- nEBP, nESP

7: blabla

6:-EIP j- CALL

5:-EBP j- yo, de buena onda

4:-llamada j- EBP, ESP

3:-

2:-

1: blabla

0: blabla

- (e) Se guarda el valor del EIP para poder continuar una vez terminada la subrutina. Y también se suele guardar el EBP

| | a | b |
|-----|--------------------------|---|
| DEC | 1 operando, r/m | decrementa en 1 su valor |
| ADD | 2 operandos, r/m e imm/r | le suma el segundo operando al valor en la posición del primero |
| MOV | 2 operandos, r/m y r/m | copia el valor del segundo operando al primer operando |
| JZ | 1 operando | Si el flag 0 está marcado (aka. el resultado de la última operación a |

- (f) Pop-ear el último valor del stack (asumiendo que es el EIP guardado) y lo guarda en el registro del EIP
- (g) Hay que asegurarse que sea el EIP guardado, se suele guardar en la posición del EBP (que a su vez suele ser la base del stack)
- (h) Puede tener un ancho de 16-bits o de 32-bits. En 64-bits tiene de 16-bits o 64-bits
- (i)

Sí, pero habría que asegurarse otra forma de ubicar el EIP y una forma de saber cuánto del stack usé. CONVENCIDO: Si está resuelto por afuera mío que no se me llene el stack, no hace falta saber cuánto stack usé

~~Galileo~~ Galileo Acuerda con Galileo

Juan Sí, pero depende del uso que necesité. (eg. no necesito volar al EIP)

4. (a) ; Calcula $9 * 3$
MOV EBX, 9 ; Valor
MOV ECX, 3 ; *3, lo uso de contador para las sumas
MOV EAX, 0 ; Empiezo en cero
loop:
ADD EAX, EBX
DEC ECX
JZ end
JMP loop
end:
MOV 0x0000000000000022, EAX ; Resultado en la posición 42 de la memoria