EJERCICIOS EC

Julio A. Fresneda - 49215154F

21.01.17

JULIO FRESNEDA

49215154F

3.1

OPERADOR	VALOR
%eax	0x100
0x104	0xAB
$0x108	0x108
(%eax)	0xFF
4(%eax)	0xAB
9(%eax, edx)	0x11
260(%ecx, %edx)	0x13
0xFC(, %eax, 4)	0xFF
(%eax, %edx, 4)	0x11

260 = 0x108

3.2

1. movl %eax, (%esp) # se usan long
2. movw (%eax), %dx # se usan word
3. movb $0xFF, %bl # byte
4. movb (%esp, %edx, 4), %dh # byte
5. pushl $0xFF # long
6. movl %dx, (%eax) # long
7. popl %edi # long

3.3

1. (%bl) no sirve
2. -l no es para %ax, debería usarse -w
3. ~~%bl~~ no se puede mover de memoria a memoria
4. De %rh no se sabe nada
5. No se puede mover nada a una cte.
6. -l no es para %dx, debería usarse -w
7. -b no es para %si.

3.4

~~char int movb %al, (%edx)~~

~~char mismos~~ ~~NO SÉ HACERLO~~

HECHO MÁS ADELANTE

3.5

```
void code(int *xp, int *yp, int *zp){
    int a = *xp;
    int b = *yp;
    int c = *zp;

    *yp = a;
    *zp = b;
    *xp = c;
}
```

3.46

AUMENTO

A) 2010 → 16'3 GB 256TB / 16'3GB = 16082'45

Hacemos 16082'45 / 1'48 hasta que el resultado sea < 1... Da aproximadamente 25 años.

$16 \cdot 10^8$ GB

B) ~~16 TB~~ / 16'3 GB = ~~[illegible]~~ 9815950012

~~AUMENTO~~ AUMENTO

Hacemos lo mismo que antes.

Aproximadamente 50 años.

3.47

int	long	movlq	%edi	%rax
char	long	movbq	%dil	%rax
uns. int	uns. long	movlq	%edi	%rax
uns. char	uns. long	movlq	%dil	%rax
~~long~~	~~int~~	~~movq~~		

3.4

char	int	movsbl	%al	(%edx)
char	unsigned	movsbl	%al	(%edx)
uns. char	int	movsbl	%al	(%edx)
int	char	movsb	%al	(%edx)
unsigned	uns. char	movzb	%al	(%edx)
unsigned	int	movl	%eax	(%edx)

3.6

leal 6(%eax), %edx	6 + x
leal (%eax, %ecx), %edx	x + y
leal (%eax, %ecx, 4), %edx	4y + x
leal 7(%eax, %eax, 8), %edx	8x + x + 7 = 9x + 7
leal 0xA(, %ecx, 4), %edx	4y + 10
leal 9(%eax, %ecx, 2), %edx	2x + x + 9

3.7

addl %ecx, (%eax)	0x100	0x100
subl %edx, 4(%eax)	0x104	0xA8
imull $16, (%eax, %edx, 4)	0x10C	0x110
incl 8(%eax)	0x108	0x14
decl %ecx	%ecx	0x0
subl %edx, %eax	%eax	0x0FD = 0xFD

3.9

```
*1 = x ^ y
*2 = *1 >> 3      // NO SÉ COMO SE PONE EN C
*3 = ~*2          // NO SÉ COMO SE PONE EN C
*4 = *3 - 2
```

3.10

A) Al hacer un X-OR en dos números iguales, el resultado queda a 0, ya que bit a bit nunca ocurre que sean bits distintos. Equivale a x = 0.

B) movl $0, %edx

C) ?

3.13

A) Al usar cuatro registros %eax, %ebx, data_t debe tener 4 bytes, como int o *char, float...

B) Al usar cuatro %ax, %dx, data_t tiene 16 bits (2 bytes), data_t es short.

C) Al usar cuatro %al, %dl, data_t tiene 8 bits (1 byte), data_t es char.

D) Igual que A).

3.14

A) Usa -l, los registros son de 32 bits. Como resta no distingue de signo. data_t puede ser int, puntero, unsigned.

B) Usa -w, los registros son de 16 bits. resta no distingue de signo.

data_t es short.

c) Se usa -b y los registros son de 8 bits. data_t sólo puede ser char.

b) Se usa -w y los registros son de 2 bytes. data_t debe ser short.

3.16

A) void code(int a, int *p){
    if(p == 0) goto fin;
    if(a <=0) goto fin;
    *p = *p + a;
fin:
}

B) Se necesitan dos condicionales porque la condición inicial (p && a > 0) son dos condiciones unidas con &&.

3.18

2. int val = x;
3. if(val < -3){
4. if(val > x){
5. val = val * y;
7. val = val + y;
8. } else if(val <= 2){
9. val = val ^ y;

3.22

```
int func a (unsigned x) {
    int val = 0;
    while (x) {
        val = val ^ x;
        x = x >> 1;
    }
    return val & 0x1;
```

3.23

```
int func_b (unsigned x) {
    int val = 0;
    int i;
    for (i = 0; i < 32; i++) {
        val = x;
        val = val << 1;
        val = val | (x & 0x1);
        x >>= 1;
    }
    return val;
}
```

3.26

El operador es "/", ya que se usa sarl $2, %eax, es decir, divide entre 4.

3.27

```
int test( int x, int y) {
    int val = 4*x;
    //int tmp = x + y;
    if ( 0 < y) {
        if ( x < y)
            val = x-y;
        else
            val = x^y;
    } else if ( y < -2)
        val = x + y;
    return val;
}
```

3.29

```
int switcher(int a, int b, int c) {
    int answer;
    switch(a) {
        case 16
            c = b ^ F;
        case 23
            answer = c + 112;
            break;
        case 12:
        case 15:
            answer = 4;
            break;
        case 12:
            answer = b;
            break;
        case 14:
            answer = (c + b) << 2;
    }
}
```

3.32

```
fun(short c, char d, int *p, int x) {
    *p = d;
    return x-c;
}
```

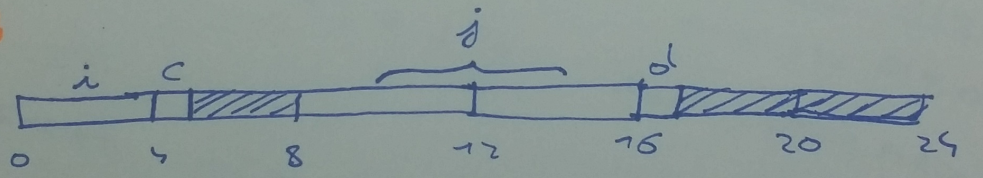
3.34

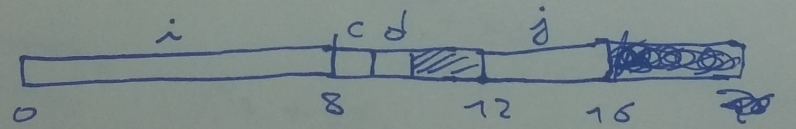
A) Guarda x.

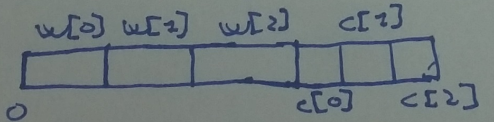
B)
```
int rfun(unsigned x) {
    if (x == 0)
        return 0;
    unsigned nx = x >> 1;
    int rv = rfun(nx);
    return x & 1 + rv;   //NO ESTOY MUY SEGURO
```

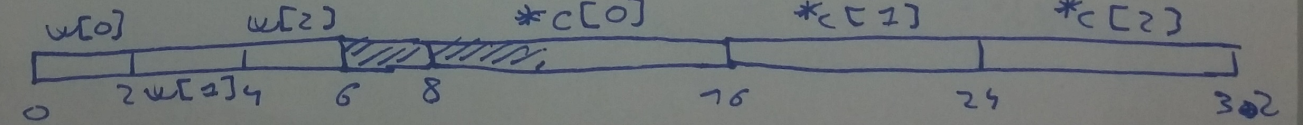
C) La función devuelve 0 si x es 0, si no es el caso, se llama a sí misma con x/2 como argumento, y retorna ese resultado (mas 1 si x == 1).

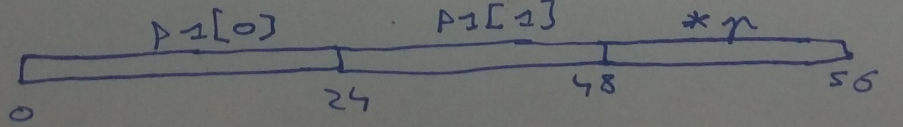
3.53

A)


B)


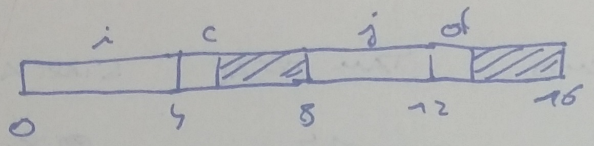
C)


D)


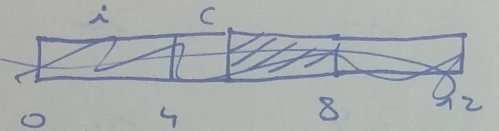
E)


3.41

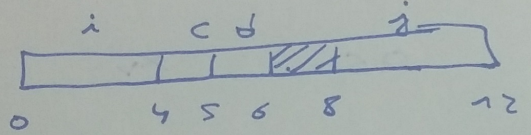
A)



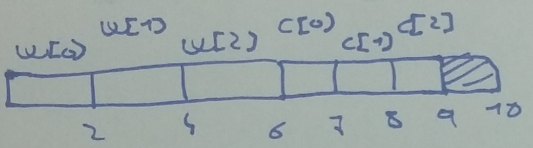
B)



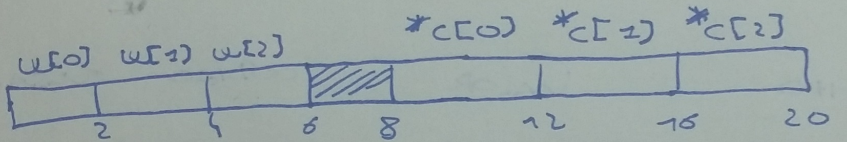
C)



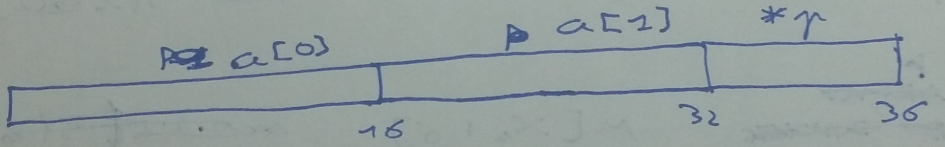
C)



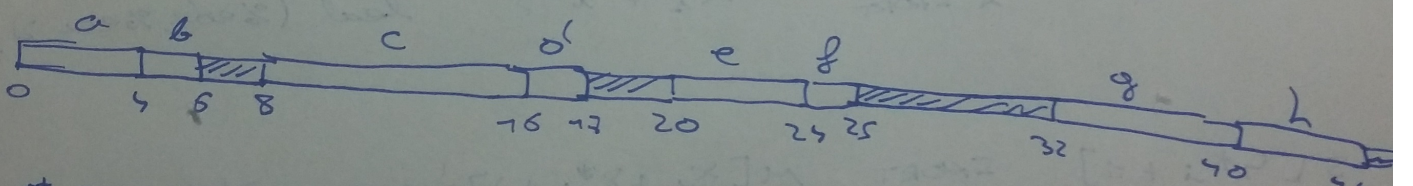
D)



E)



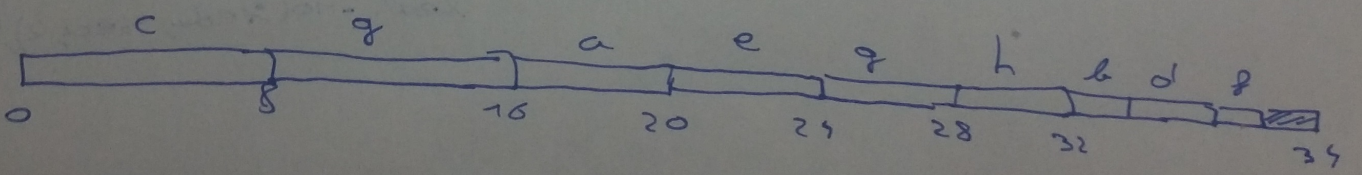
3.42



A) * SE VE EN DIBUJO

B) 48 ~~BYTES~~ BYTES

C)



TOTAL: 34 ~~BYTES~~ BYTES

3.35

SUPONGO 32 BITS

ARRAY	TAM ELEMENTO	TAM TOTAL	START A.	ELEM. i
S	2	14	X_s	$X_s + 2i$
T	4	12	X_T	$X_T + 4i$
U	4	24	X_U	$X_U + 4i$
V	12	96	X_V	$X_V + 12i$
W	4	16	X_W	$X_W + 4i$

3.36

EXPRESION	TIPO	VALOR	CODIGO EJECUTABLE
S + 1	PUNTERO A SHORT	$X_s + 2$	leal 4(%edx), %eax
S[3]	SHORT	$M[X_s + 6]$	movw 6(%edx), %ax
&S[i]	PUNTERO A SHORT	$X_s + i*2$	leal (%edx, %ecx, 2), %eax
S[4*i + 1]	SHORT	$M[X_s + 8*i + 2]$	movw 2(%edx, %ecx, 8), %ax
S+i-5	PUNTERO A SHORT	$X_s + 2*i - 10$	leal -10(%edx, %ecx, 2), %eax