Edgar Aguadé Nadal.

Ex 3.34 8

```
char x;
int z;
char w[20];

char A(char i, int k, char *v) {
    int r[10];
    char c;
    c = v[3];
    r[k] = i;
    return c;
}
```

a)

```
la $t0, x
lb $a0, 0($t0) # param x
la $t0, z
lw $a1, 0($t0) # param z.
la $a2, w      # param *w.

jal A #crida subrutina.

la $t0, x # estava a $t0 per ser temporal
lb $v0, 0($t0) # x = A(x, z, w);
```

b) Bloc d'Activació 8

r[0]	← 0($sp)
r[1]	
r[2]	
r[3]	
r[4]	
r[5]	
r[6]	
r[7]	
r[8]	
r[9]	Total: 40 bytes.

c)

```
addiu $sp, $sp, -40 #reserva BoA.
lb $t0, 3($a2)   # c = v[3]
sll $t1, $a1, 2  # k*4
addu $t1, $t1, $sp  # &r[k]
sw $a0, 0($t1)   # r[k] = i
move $v0, $t0    # return c
addiu $sp, $sp, 40 # Allibera BoA.
jr $ra.
```

Ex 3.26 8

```
char v[7];
int s3 (char v1[], int tf, int nf, int/char c);

int subr3(int param) {
    int ret, tam = 7;

    ret = s3(v, tam, param, 'D');
    $v0
    return ret + param + tam;
}
```

$s0 ← param
$s1 ← tam
$ra
} 12 bytes

```
addiu $sp, $sp, -12  #Reserva BoA
sw $s0, 0($sp)       # $s0 old
sw $s1, 4($sp)       # $s1 old
sw $ra, 8($sp)       # $ra
move $s0, $a0        # param → segur
li $s1, 7            # let tam = 7 → segur.
la $a0, v            # &v
move $s1, $a1        # tam
move $s0, $a2        # param
li $a3, 'D'          # 'D'
jal s3               # crida.
...
addu $v0, $v0, $s0   } prep return.
addu $v0, $v0, $s1   }

lw $s0, 0($sp)       } recup.
lw $s1, 4($sp)       } old
lw $ra, 8($sp)       } valors.

addiu $sp, $sp, 12
jr $ra # return.
```

Ex 3.36 8

```
void sub(int p1[10], int p2, int p3){
    int x1[10], x2;
    ...
    x1[p2] = p1[x2] + p3;
    ...
    sub(x1, p2, p1[3]);
    ...
    x2 = p2 + p3;
}
```

*Al tenir els punts suspensius no podem asumir que allà no necesitem altres valors passats, per tant guardo tot al BoA.

p1[] → $s0 (el punter al primer elem.)
p2 → $s1
p3 → $s2
x1[10] → stack
x2 → $s3

0($sp)	$ra	4 byte
4($sp)	$s0	4 byte
8($sp)	$s1	4 byte
12($sp)	$s2	4 byte
16($sp)	$s3	4 byte
20($sp)	x1 ⋮	40 byte.

Total: 60 bytes

sub8

```
addiu $sp, $sp, -60
sw $ra, 0($sp)
sw $s0, 4($sp)
sw $s1, 8($sp)
sw $s2, 12($sp)
sw $s3, 16($sp)

move $s0, $a0  # &p1[]
move $s1, $a1  # p2
move $s2, $a2  # p3
move $s3, $zero # int x2;
...
sll $t0, $s3, 2 # x2*4
addu $t0, $s0, $t0 # p1[x2]
addu $t0, $t0, $s2 # p1[x2]+p3
addiu $t1, $sp, 20 # &x1[0]
sll $t2, $s1, 2   # p2*4
addu $t1, $t1, $t2 #&x1[p2]
sw $t0, 0($t1)  # x1[p2] = p1[x2]+p3
...
addiu $a0, $sp, 20 # x1
move $a1, $s1  # p2
addiu $a2, $s0, 12 # p1[3]
jal sub
...
addu $s3, $s1, $s2 # x2 = p2+p3
...

lw $ra, 0($sp)
lw $s0, 4($sp)
lw $s1, 8($sp)
lw $s2, 12($sp)
lw $s3, 16($sp)

addiu $sp, $sp, 60
jr $ra # surt de sub.
```