

ΗΥ 232  
Οργάνωση και  
Σχεδίαση Υπολογιστών

Διάλεξη 6

Αναδρομή

Νίκος Μπέλλας  
Τμήμα Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων

# Υπολογισμός Παραγοντικού n!

$$n! = 1*2*..(n-1)*n = \begin{cases} \text{if } n < 1, 1 \\ \text{else, } n*(n-1)! \end{cases}$$

```
int fact (int n)
{
    if (n < 1) return 1;
    else return n * fact(n - 1);
}
```

- Παράμετρος n στον \$a0
- Αποτέλεσμα στον \$v0

# Ένθετες διαδικασίες: Παράδειγμα

jal fact

LLL:

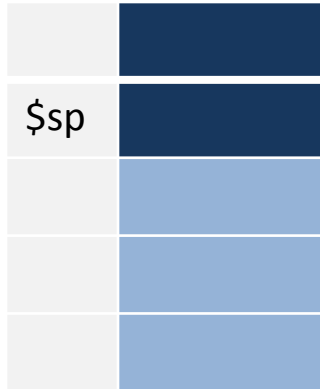
```
int fact (int n)
{
    if (n < 1) return 1;
    else return n * fact(n - 1);
}
```

fact:		
	addi \$sp, \$sp, -8	# ρύθμιση στοίβας για 2 τιμές
	sw \$ra, 4(\$sp)	# αποθ. διεύθυνσης επιστροφής
	sw \$a0, 0(\$sp)	# αποθ. παραμέτρου
	bge \$a0, 1, L1	# έλεγχος για το εάν n<1
	li \$v0, 1	# αν ναι, αποτέλεσμα: 1
	addi \$sp, \$sp, 8	# αφαιρ. 2 τιμών από στοίβα
	jr \$ra	# και επιστροφή
L1:	addi \$a0, \$a0, -1	# αλλιώς αφαιρ. 1 από το n
	jal fact	# αναδρομική κλήση
LL:	lw \$a0, 0(\$sp)	# επαναφορά αρχικού n
	lw \$ra, 4(\$sp)	# και διεύθυνσης επιστροφής
	addi \$sp, \$sp, 8	# αφαιρ. 2 τιμών από στοίβα
	mul \$v0, \$a0, \$v0	# υπολογ. αποτελέσματος
	jr \$ra	# και επιστροφή

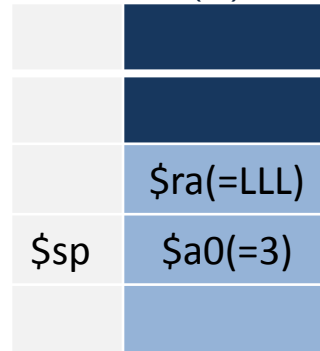
# Συμπεριφορά της στοίβας

Έστω ότι η συνάρτηση καλείται με  $n=3$

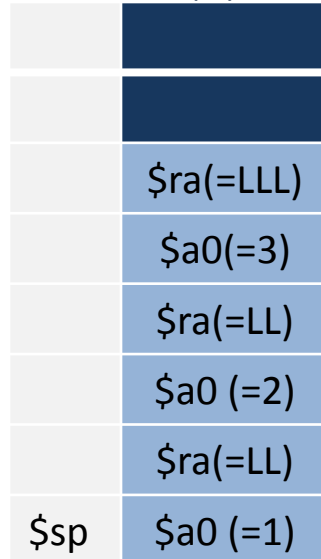
Πριν την κλήση της fact(3)



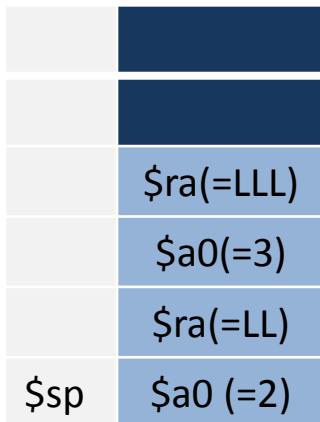
fact(3)



fact(1)



fact(2)



fact:

```
addi $sp, $sp, -8 # ρύθμιση στοίβας
sw $ra, 4($sp) # αποθ. διεύθυνσης
sw $a0, 0($sp) # αποθ. Παραμέτρου
```

```
bge $a0, 1, L1 # έλεγχος για το ε
```

```
li $v0, 1 # αν ναι, αποτέλεσμα
addi $sp, $sp, 8 # αφαιρ. 2 τιμών απ
jr $ra # και επιστροφή
```

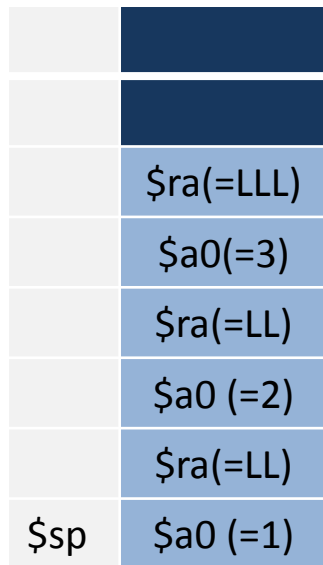
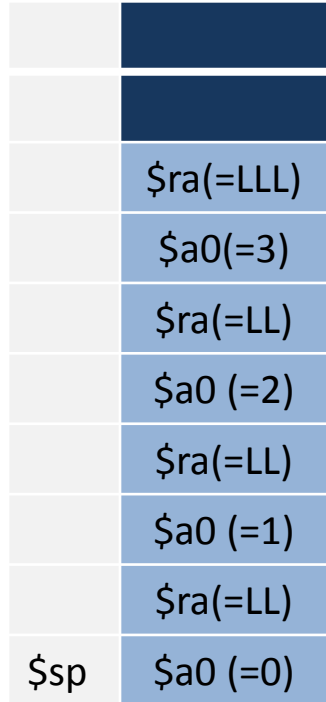
```
L1: addi $a0, $a0, -1 # αλλιώς αφαιρ. 1
jal fact # αναδρομική κλήση
```

```
LL: lw $a0, 0($sp) # επαναφορά αρχικο
lw $ra, 4($sp) # και διεύθυνσης επ
addi $sp, $sp, 8 # αφαιρ. 2 τιμών απ
```

```
mul $v0, $a0, $v0 # υπολογ. αποτελέσ
jr $ra # και επιστροφή
```

# Συμπεριφορά της στοίβας (συνεχ.)

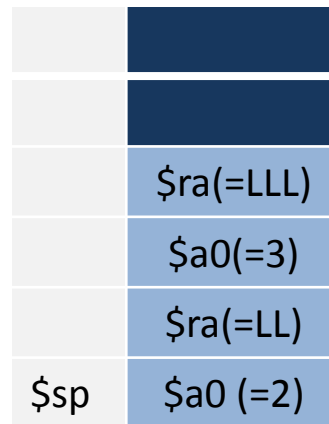
fact(0)



$\$v0 = 1$



$\$v0 = 2*3=6$



$\$v0 = 1*1=1$

```
fact:
    addi $sp, $sp, -8    # ρύθμιση στοίβας
    sw   $ra, 4($sp)    # αποθ. διεύθ. $ra
    sw   $a0, 0($sp)    # αποθ. Παράμ. $a0

    bge  $a0, 1, L1     # έλεγχος για 0

    li   $v0, 1         # αν ναι, αποθ. αποτέλεσμα
    addi $sp, $sp, 8    # αφαιρ. 2 τιμές
    jr   $ra            # και επιστροφή

L1:    addi $a0, $a0, -1 # αλλιώς αφαιρ. 1
       jal  fact        # αναδρομική

LL:    lw   $a0, 0($sp) # επαναφορά $a0
       lw   $ra, 4($sp) # και διεύθ. $ra
       addi $sp, $sp, 8 # αφαιρ. 2 τιμές

    mul  $v0, $a0, $v0  # υπολογ. αποτελ.
    jr   $ra            # και επιστροφή
```

Επέστρεψε στην καλούσα συνάρτηση

$\$v0 = 1*2=2$

# Παράδειγμα αναδρομής

- Ισχυρή αναδρομική συνάρτηση *tak()*

```
int tak(int x, int y, int z) {
    if (y < x) {
        return 1 + tak(tak(x-1,y,z),
                       tak(y-1,z,x),
                       tak(z-1,x,y));
    }
    else
        return z;
}

int main () {
    tak (18, 12, 6);
}
```

# Παράδειγμα αναδρομής

```
int tak(int x, int y, int z) {
    if (y < x) {
        return 1 + tak(tak(x-1,y,z),
                       tak(y-1,z,x),
                       tak(z-1,x,y));
    }
    else
        return z;
}

int main () {
    tak (18, 12, 6);
}
```

tak:

```
subu   $sp, $sp, 20
sw     $ra, 16($sp)
sw     $s0, 12($sp)
sw     $s1, 8($sp)
sw     $s2, 4($sp)
sw     $s3, 0($sp)
move   $s0, $a0
move   $s1, $a1
move   $s2, $a2
```

```
# θέσεις για 5 καταχωρητές στην
# στοίβα: $ra, $s0..$s3
```

```
# $s0 = x
# $s1 = y
# $s2 = z
```

```
bge    $s1, $s0, L1
```

```
# if (y >= x) goto L1
```

```
addiu  $a0, $s0, -1
move   $a1, $s1
move   $a2, $s2
jal    tak
move   $s3, $v0
```

```
# κώδικας για y < x
```

```
# $s3= tak(x-1, y, z)
```

# Παράδειγμα αναδρομής

```
int tak(int x, int y, int z) {
    if (y < x) {
        return 1 + tak(tak(x-1,y,z),
                       tak(y-1,z,x),
                       tak(z-1,x,y));
    }
    else
        return z;
}

int main () {
    tak (18, 12, 6);
}
```

```
addiu $a0, $s1, -1
move  $a1, $s2
move  $a2, $s0
jal   tak
```

```
# $v0=tak(y-1, z, x)
```

```
addiu $a0, $s2, -1
move  $a1, $s0
move  $a2, $s1
move  $s0, $v0
jal   tak
```

```
# $v0=tak(z-1, x, y)
```

```
move  $a0, $s3
move  $a1, $s0
move  $a2, $v0
jal   tak
addiu $v0, $v0, 1
j     L2
```

```
#tak (tak(..), tak(..), tak(..))
```

```
L1:  move  $v0, $s2
```



# Παράδειγμα αναδρομής

```
int tak(int x, int y, int z) {
    if (y < x) {
        return 1 + tak(tak(x-1,y,z),
                       tak(y-1,z,x),
                       tak(z-1,x,y));
    }
    else
        return z;
}

int main () {
    tak (18, 12, 6);
}
```

```
L2:   lw    $ra, 16($sp)
      lw    $s0, 12($sp)
      lw    $s1, 8($sp)
      lw    $s2, 4($sp)
      lw    $s3, 0($sp)
      addiu $sp, $sp, 20
      jr    $ra
```

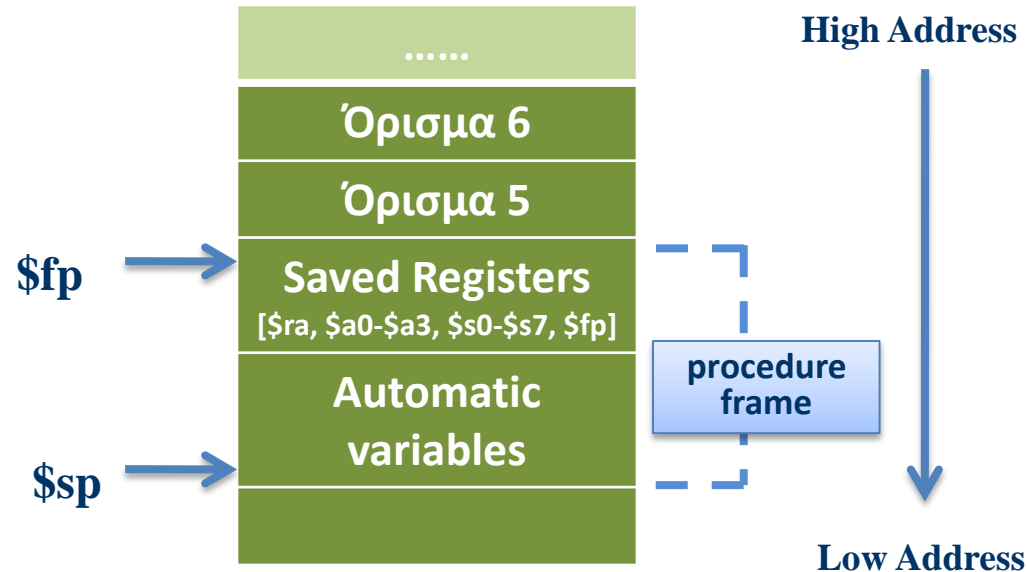
main:

```
      li    $a0, 18
      li    $a1, 12
      li    $a2, 6
      jal   tak
      # προετοιμασία και κλήση της tak()

      move  $a0, $v0
      li    $v0, 1
      syscall
      # τύπωσε το αποτέλεσμα
```

# Καταχωρητής Πλαισίου (\$fp)

- Η προσπέλαση των αυτόματων μεταβλητών στη στοίβα δεν γίνεται εύκολα μέσω του \$sp, επειδή αυτός μεταβάλλεται διαρκώς όσο προστίθενται νέες και αφαιρούνται παλαιές μεταβλητές  
=> συνήθως μια διαδικασία ορίζει το χώρο που καταλαμβάνουν οι σωσμένοι καταχωρητές και οι αυτόματες μεταβλητές ως το χώρο *πλαίσιο* της (procedure frame) στη στοίβα, και μαρκάρει τη *σταθερή* αρχή αυτού με τον καταχωρητή \$fp (frame pointer, \$30)



# Τοπικές μεταβλητές διαδικασιών

- Προσοχή. Μια διαδικασία δεν μπορεί να επιστρέψει στο καλούν πρόγραμμα έναν pointer σε αυτόματη μεταβλητή που έχει δημιουργηθεί στη στοίβα (π.χ. σε έναν πίνακα, όπως συχνά συμβαίνει), καθώς ο χώρος αυτός της μνήμης χάνεται με την επιστροφή από τη διαδικασία και το ταυτόχρονο pop όλων των αυτόματων μεταβλητών

```
int *f() { // error !
    int local[10];

    for (i=0;i<10;i++)
        local[i] = i;
    return &local[0];
}
```

=> η επιστροφή ενός πίνακα μπορεί να γίνει μόνο με τη μέθοδο της δέσμευσης χώρου στη μνήμη από το καλούν πρόγραμμα, και το πέρασμα της αντίστοιχης διεύθυνσης (pointer) βάσης στους καταχωρητές \$a0-\$a3 ως παραμέτρου της διαδικασίας