Muito breve resumo sobre estruturas

Tomás Oliveira e Silva, Dezembro de 2009

Código C	Alinhamento (múltiplo de)	Tamanho	Posição (offset)
`typedef struct`			
`{`			
`  char v1;`	1	1	0
`  int v2;`	4	4	$\geq 0+1 \rightarrow 4$
`  char v3;`	1	1	$\geq 4+4 \rightarrow 8$
`  int v4[10];`	4	10×4	$\geq 8+1 \rightarrow 12$
`  char v5[10];`	1	10×1	$\geq 12+40 \rightarrow 52$
`  float v6;`	4	4	$\geq 52+10 \rightarrow 64$
`  char v7;`	1	1	$\geq 64+4 \rightarrow 68$
`}`			
`tipo;`			
	4 (o maior)	$\geq 68+1 \rightarrow 72$	

Código C	Assembly MIPS

`static tipo x[3];`	`.data`
	`.align 2`
	`x: .space 216  # 3*72`

`int i,j,k; // $t0,$t1,$t2`
`char c,*s; // $t3,$t4`
`float f;  // $f4`
`tipo *t;  // $t5`	

`t = &x[i];`
`  // t->`
`  //  passa a ser o mesmo que`
`  // x[i].`
	`la $t5,x`
	`li $t6,72`
	`mult $t6,$t0,$t6  # 72*i`
	`addu $t5,$t5,$t6  # &x[i]`

`c = x[i].v1;` `c = t->v1;`	`lb $t3,0($t5)  # offset=0` `lb $t3,0($t5)  # offset=0`
`k = x[i].v2;` `k = t->v2;`	`lw $t2,4($t5)  # offset=4` `lw $t2,4($t5)  # offset=4`
`c = x[i].v3;`	`lb $t3,8($t5)  # offset=8`

`k = x[i].v4[j];`	`sll $t6,$t1,2  # 4*j`
	`addu $t6,$t5,$t6  # (ver nota)`
	`lw $t2,12($t6)  # offset=12`

`s = x[i].v5;` `s = t->v5;`	`addiu $t4,$t5,52  # offset=52` `addiu $t4,$t5,52  # offset=52`
`s = &x[i].v5[1];`	`addiu $t4,$t5,53  # offset=52+1`
`c = t->v5[3];`	`lb $t3,55($t5)  # offset=52+3`
`f = x[i].v6;`	`l.s $f4,64($t5)  # offset=64`
`x[i].v7 = c;` `t->v7 = c;`	`sb $t3,68($t5)  # offset=68` `sb $t3,68($t5)  # offset=68`

`static tipo y = {`
`  'X',`
 	
`  17,`
`  'Y',`
 	
`  { 1,2,3 },`
 	
`  "ABC",`
 	
`  3.14,`
`  'Z'`
 	
`};`
	`.data`
	`.align 2`
	`y: .byte 'X'`
	`.space 3  # optional`
	`.word 17`
	`.byte 'Y'`
	`.space 3  # optional`
	`.word 1,2,3`
	`.space 28  # 7*4 bytes`
	`.asciiz "ABC"  # 4 bytes`
	`.space 8  # 6+2 bytes`
	`.float 3.14`
	`.byte 'Z'`
	`.space 3  # optional`

Nota: o endereço de `x[i].v4[j]` é, já em bytes, dado por `x+72*i+12+4*j`; `$t6` fica com `x+72*i+4*j` pelo que só falta somar 12, o que é feito na instrução `lw`.