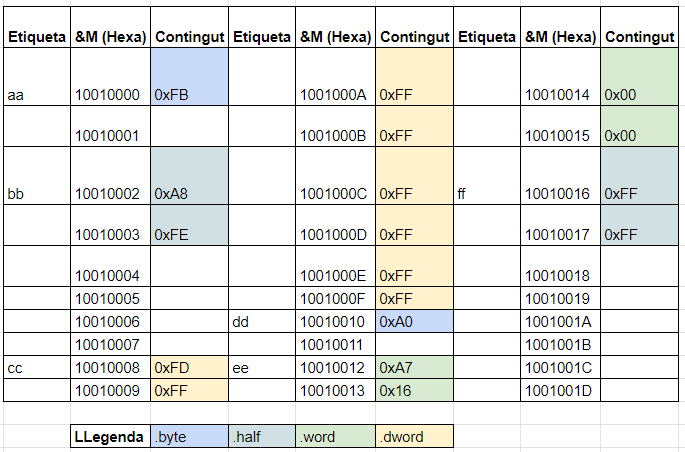
# Pràctica 1

## Activitat 1.A

### Exercici 1.1

|  |
| --- |
| .data  aa: .byte -5  bb: .half -344  cc: .dword -3  dd: .byte 0xA0  # Tot i .byte no distingeix entre signed i unsigned                  # Hauriem de fer això en el .text: li $t0, 0xA0 && sb $t0, 0(dd)  ee: .word 5799  ff: .half -1 |

### Exercici 1.2



## Activitat 1.B

|  |  |
| --- | --- |
| **Macros MIPS** | **Instruccions MIPS** |
| la $s3, aa | lui $1,0x00001001  ori $19,$1,0x00000000 |
| li $s4, 65535 | ori $20,$0,0x0000FFFF |
| li $s5, 65536 | lui $1,0x00000001  ori $21,$1,0x00000000 |
| move $s0, $s1 | addu $16,$0,$17 |

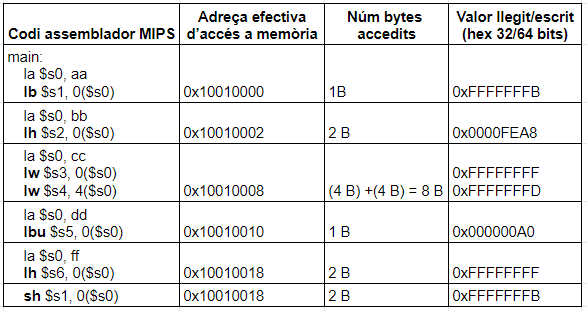
## Activitat 1.C

### Exercici 1.3

// sh és “store half” i aquest escriu 2 B = 16 b. En el cas concret, fa “M[$s0] = $s1” 🡪 “ff = aa”.

// Tot i que lb,lh,… no llegeixin tot el contingut, si aquest és negatiu hem de fer extensió de signe.

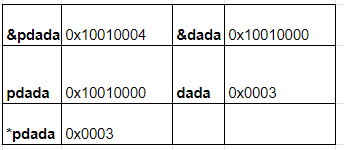
// Com que treballem Little-Endian primer llegim la part superior “FFFF” i després la inferior “FFFD”.



## Activitat 1.D

### Exercici 1.4

// És important dir que en aquest cas teòric “\*pdada” serà “0x0003” donat que está apuntant a la direcció de “dada” i “dada” té contingut 0x0003 donat que és un “.half”, però les variables en MARS son de 64 bits i si carregues en una d’aquestes el punter “\*pdada” es veu tal que “0x00000003”

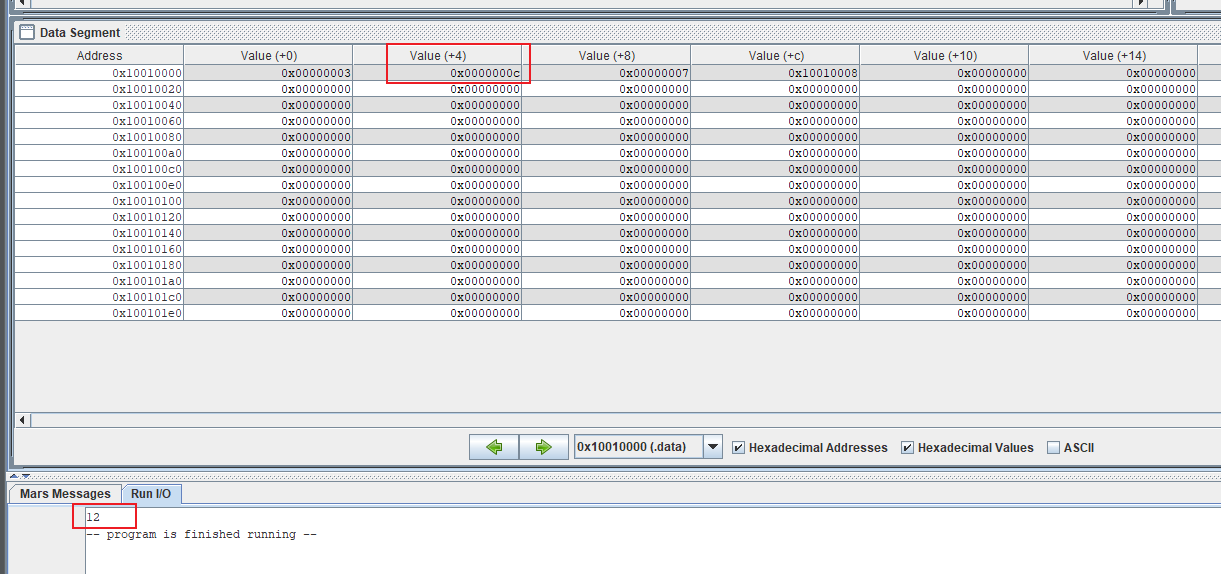


## Activitat 1.E

### Exercici 1.5

// No declaro “temp” en “.data” perquè en el codi, “temp”, és local (Dins del main).

|  |
| --- |
| .data  A:              .word 3,5,7  # A[3] = {3,5,7};  punter:         .word 0  # punter = 0;  .text  .globl main  main:          la $t0,A  # $t0 = &A[0]          addiu $t0,$t0,8  # $t0 = &A[0+2]  ## 8=2\*4 on 2 es la pos. i 4 pq. es word          la $t1,punter  # $t1 = &punter          sw $t0,0($t1)  # punter = &A[2];            lw $t2,0($t1)  # $t2 = punter ($t1)  ## Valor de punter (que es &A[2])          lw $t3,0($t2)  # temp = A[2]  ## temp es la variable $t3          addiu $t3,$t3,2  # temp = temp + 2;          lw $t2,0($t1)  # $t2 = punter          lw $t4,-8($t2)  # temp2 = A[0]  ## tempo2 es la variable $t4          addu $t3,$t4,$t3  # temp = temp2 + temp;            lw $t2,0($t1)  # $t2 = punter          sw $t3,-4($t2)  # A[1] = temp;          li $v0,1  # Carregar el codi del syscall per fer print          move $a0,$t3  # Mou a $a0 (Variable del MARS) el valor de temp ($t3).          syscall  # Executa el print(temp)          jr $ra          # main retorna al codi de startup |



## Activitat 1.F

### Exercici 1.6

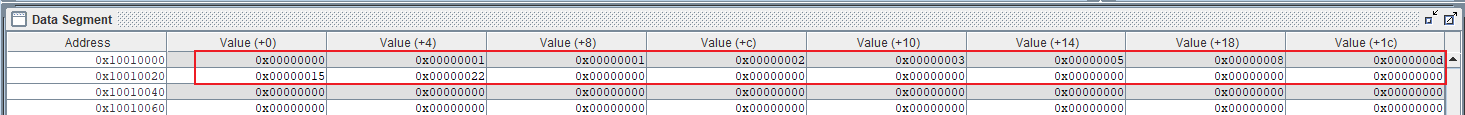
// No guardem el valor a una dirección de memoria, sino que ho movem de registre a registre, per això no necessitem fer cap “sw”.

|  |
| --- |
| .data  vec:    .word 9,8,7,6,5,4,3,2,1  # &vec[i] = &vec[0]+(tamany\_element\_vector(4)\*posicio(i))  .text  .globl main  main:      # OBJECTIU:  $s1 <- vec[i]      la $t0,vec  # $t0 = @vec[0]  ## Guardar valor primera pos del vector      sll $t1,$s2,2  # $t1 = $s2\*4  ## Multiplicar per 4 el valor de 'i' (que esta en $s2)      addu $t0,$t0,$t1 # $t0 = $t0+$t1 ## Sumar la direccio final      lw $s1,0($t0)  # $s1 = vec[2]  ## Carregar el contingut de la pos 2 del vector      jr $ra  # Tornar al inici |

## Activitat 1.G

### Exercici 1.7

|  |
| --- |
| .data  fib:    .space 40 # 4\*10        .text      .globl main  main:      li $s0,2  # i = 2; ## $s0 <-> i      la $t2,fib      li $t0,0      sw $t0,0($t2) # fib[0] = 0;      li $t0,1      sw $t0,4($t2) # fib[1] = 1;    while:      slti $t0,$s0,10      beq $t0,$zero,fi  # while (i<10)        la $t3,fib      sll $t0,$s0,2      addu $t3,$t3,$t0  # $t3 = @fib[i]      lw $t1,-4($t3)  # $t1 = fib[i-1]      lw $t2,-8($t3)  # $t2 = fib[i-2]      addu $t4,$t1,$t2  # $t4 = fib[i-1] + fib[i-2]      sw $t4,0($t3)  # fib[i] = $t4        addiu $s0,$s0,1  # i++;      b while  fi: |



## Activitat 1.H

### Exercici 1.8

|  |
| --- |
| .data  cadena: .byte -1,-1,-1,-1,-1,-1  vec:    .word 5,6,8,9,1        .text      .globl main  main:      li $s0,0  # i = 0; ## $s0 <-> i  while:      li $t0,5  # $t0 = 5      bge $s0, $t0, fi  # while(i<=5)        la $t5,vec      li $t4,4  # $t4 = 4      subu $t2,$t4,$s0  # $t2 = 4-i      sll $t2,$t2,2      addu $t3,$t5,$t2  # $t3 = @vec[4-i]      lw $t2,0($t3)  # $t2 = vec[4-i]      li $t3,'0'  # $t3 = 0x30      addu $t4,$t2,$t3  # $t4 = vec[4-i]+'0'      la $t0,cadena      addu $t0,$t0,$s0  # $t0 = @cadena[i]      sb $t4,0($t0)  # cadena[i] = $t4  ## CUIDADO QUE HA DE SER sb perque cadena es de bytes        addiu $s0,$s0,1  # i++;      b while  fi:      la $t0,cadena      sb $zero,5($t0)  # cadena[5] = 0 ## Donat que es byte, puc ficar el 5 directament        li $v0,4 # Fer print\_string      move $a0,$t0      syscall        jr $ra  # Tornar al inici |

