



1. Introdução

Pretende-se com esta aula prática que os alunos se familiarizem com o formato de instruções do MIPS32, convertendo um programa de *assembly* para código máquina.

2. Conversão para binário

Considere o seguinte programa, escrito em C:

```
prog.c
typedef struct {
    char nome[10];
    int idade; } Telem;

Telem membros[10];

main ()
{
    int i, total;

    total = 0;
    for (i=0 ; i< 10 ; i++)
        total += membros[i].idade;
    printf ("%d\n", total);
}
```

Uma vez compilado usando o comando

```
mips-gcc -S -O1 -mrnames prog.c
```

gera o código *assembly* apresentado na próxima tabela.

```

                                prog.s
$LC0: .ascii    "%d\n\000"
main:
    subu    $sp,$sp,24          #####
    sw     $ra,16($sp)         #####
    move   $a1,$zero           #####
    move   $v1,$zero
    la     $a0,membros         #####
    sll    $v0,$v1,4           #####
$L9:
    addu   $v0,$v0,$a0
    lw     $v0,12($v0)         #####
    nop
    addu   $a1,$a1,$v0
    addu   $v1,$v1,1
    slt   $v0,$v1,10          #####
    .set   noreorder
    .set   nomacro
    bne   $v0,$zero,$L9       #####
    sll   $v0,$v1,4
    .set   macro
    .set   reorder

    lui   $a0,%hi($LC0) # high
    .set   noreorder
    .set   nomacro
    jal   printf
    addiu $a0,$a0,%lo($LC0) # low
    .set   macro
    .set   reorder

    lw     $ra,16($sp)
    nop
    .set   noreorder
    .set   nomacro
    j     $ra
    addu   $sp,$sp,24
    .set   macro
    .set   reorder
    .end main
    .comm membros,160

```

Questão 1 – Passe para código máquina as instruções assinaladas com #### , sabendo que:

- `subu`, `move`, `la` são pseudo-instruções;
- algumas instruções com valores imediatos aparecem com a mnemónica da instrução correspondente do tipo R, mas o opcode apropriado é do tipo I (exemplo: `slt` versus `slti`).
- `nop` consiste numa sequência de 32 bits a 0;
- os endereços de membros, `$LC0` e `printf` não são conhecidos em tempo de compilação; devem ser considerados `0x00000000`, com 32 bits de tamanho;
- os números dos registos são dados na próxima tabela:

Nome	Número	Utilização	Preservado na chamada?
<code>\$zero</code>	0	Constante 0	n.a.
<code>\$v0-\$v1</code>	2-3	Valores para resultados e avaliação de expressões	Não
<code>\$a0-\$a3</code>	4-7	Argumentos	Sim
<code>\$t0-\$t7</code>	8-15	Temporários	Não
<code>\$s0-\$s7</code>	16-23	Seguros	Sim
<code>\$t8-\$t9</code>	24-25	Mais temporários	Não
<code>\$gp</code>	28	Apontador global	Sim
<code>\$sp</code>	29	Apontador para pilha	Sim
<code>\$fp</code>	30	Apontador para a <i>frame</i>	Sim
<code>\$ra</code>	31	Endereço de retorno	Sim

Verifique as suas respostas usando os comandos

```
mips-gcc -O1 -c prog.c
mips-objdump -d prog.o
```

```
00000000 <main>:
 0: 27bdffe8    addiu sp,sp,-24
 4: afbf0010    sw    ra,16(sp)
 8: 00002821    move  a1,zero
c: 00001821    move  v1,zero
10: 3c020000    lui   v0,0x0
14: 24440000    addiu a0,v0,0
18: 00031100    sll   v0,v1,0x4
1c: 00441021    addu  v0,v0,a0
20: 8c42000c    lw    v0,12(v0)
24: 00000000    nop
28: 00a22821    addu  a1,a1,v0
2c: 24630001    addiu v1,v1,1
30: 2862000a    slti  v0,v1,10
34: 1440fff9    bnez  v0,1c <main+0x1c>
38: 00031100    sll   v0,v1,0x4
3c: 3c040000    lui   a0,0x0
40: 0c000000    jal   0 <main>
44: 24840000    addiu a0,a0,0
48: 8fbf0010    lw    ra,16(sp)
4c: 00000000    nop
50: 03e00008    jr    ra
54: 27bd0018    addiu sp,sp,24
```