Lista 2 – Assembly MIPS32 (MIPS-3000)

1. Explique a diferença entre as instruções **`add`** e **`addi`**. Qual formato de instrução cada uma delas utiliza?
2. Quantos bits a instrução **`j`** reserva para o endereço do jump? Qual o formato da instrução?
3. Para que servem as instruções **`jal`** e **`jr`**? Em que contexto elas são indispensáveis?
4. Descreva a utilização dada para os seguintes registradores:

$s0 - $s7: _____

$t0 - $t7: _____

$pc: _____

$sp: _____

$ra: _____

$gp: _____

$fp: _____

$zero: _____

$a0-a3: _____

$v0-v1: _____

$hi: _____

$lo: _____

5. Converta as parcelas de código C abaixo para assembly do MIPS3000. Especifique quais variáveis serão atribuidas a quais registradores.

```
long a, b, c, d, e, f, g, h;

a) a = 0;

b) b = 42;

c) a = (b+c) / (c-d);

d) a = (a+b+c+d)/(e-f-g-h);

e) a = a/b + c/d + e*f - g*h;

f) a = 42 + b - (54 * a);
```

```
unsigned long a, b, c, d, e, f, g, h;

g) a=((b+c)*(d-e)/(f*g*h))/a+b+c+d+e;

h) a = 0xFFFFFFFA;

i) a = (a + b + d + e) / (b*b*b);

j) h = g*i + h*h - c/c;
```

```
k) a = (~b & c) | (b & ~c);

l) a = b >> 2;

m) c = ~(d << 4);

n) a = a^b^c^d;
```

6. Descreva o que significa os acronismos que definem as seguintes instruções:

sll :_____

srl :_____

slt :_____

slti :_____

beq :_____

bne :_____

beqz :_____

bgtz :_____

blez :_____

bltz :_____

j :_____

jal :_____

jr :_____

7. Converta as parcelas de código C abaixo para assembly do MIPS3000.

```
if (c < 42) {...}
```

```
if (c < d) {...}
```

```
if ((c >=0) && (c <= 42) ) {...}
```

```
if (a > 0) {...} else {...}
```

```
if (a < 0) {...}
else if (a < - 42) {...} else {...}
```

```
long c = 0;
while (c < 42)
{
   c++;
}
```

```
for (long c = 0; c < 42; c++) {...}
```

```
for (long c = 42; c >=0; --c) {...}
```

```
switch(expression){
    case 42 :
       ...
       break;
    case 31415:
       ...
       break;
    default :
       ...
}
```

8. Explique a diferença entre uma instrução nativa e uma pseudo-instrução. Dê exemplos.
9. Para que servem as instruções LI e LA. Forneça exemplos de suas utilizações.
10.
11. Converta os programas abaixo para sub-rotinas em assembly. Considere que os registradores utilizados que não forem temporários devem ser salvos. O mesmo vale para o $ra, $a0-$a3 e $v0-$v1. Explique o que as funções fazem.

```
long m(long a, long b){
  long i, r = 0;
  for(i=a; i>0; i--){
    r += b;
  }
  return r;
}
```

```
int e(int b, int exp){
  int i, r = b;
  for(i=exp; i>1; i--){
    r = m(r,b);
  }
  return r;
}
```

12. Sabemos que a instrução **j** utiliza o tipo de instrução **Type-J**, que reserva 6 bits para o opcode e os 26 bits restantes para o endereço de memória para o qual deve ser executado o salto. Qual a faixa de endereços endereçáveis utilizando a instrução **j**? Caso seja necessário saltar para uma posição de memória que esteja fora do limite imposto pelos 26 bits de **j,** o que poderíamos fazer? Considere que desejamos saltar para uma posição na faixa 0x04000000 – 0xFFFFFFFF.
13. Considere o programa abaixo. Ele multiplica uma série de 4 números (a1xb1, a2xb2, a3xb3 e a4xb4) por fim, soma os resultados das multiplicações e então imprime seu resultado. No exercício anterior fizemos um algoritmo de multiplicação através de somas sucessivas.

```
void main(){
  long i, r1,r2,r3,r4;
  long a1=1,a2=3,a3=5,a4=7;
  long b1=2,b2=4,b3=8,b4=16;

  r1=r2=r3=r4 = 0;
  //----secao----//
  for(i=a1; i>0; i--){
    r1 += b1;}
  for(i=a2; i>0; i--){
    r2 += b2;}
  for(i=a3; i>0; i--){
    r3 += b3;}
  for(i=a4; i>0; i--){
    r4 += b4;}
  //-------------//
  int r = r1+r2+r3+r4;
  printf("%d\n", r);
}
```

   a) Converta o programa acima para assembly plano sem a utilização de subrotinas e conte quantas operações são necessárias para a sua execução.
   b) Substitua a parte do programa salientada pelos comentários "seção" por quatro chamadas à subrotina "m" do exercício 5 e então conte quantas operações serão necessárias para a execução do programa.
   c) Que conclusões você tira da diferença de números de instruções a serem executadas nos dois casos? Quando esta diferença é justificada?
14. Como converteríamos as seguintes estruturas de C para assembly?

```
i=0;
while(i<N){
//faça algo útil
i = i+1;}
```

```
for(int i =0; i< N; i++){
//faça algo útil
}
```