Semana 2

Cap 1

1.1

a)
Compiler - o compilador é um “programa” que efetua a tradução de um código escrito por um ser humano numa linguagem (c por exemplo) e o converte em linguagem máquina (a linguagem do computador).

b)
Interpreter - na sequência do trabalho feito pelo compilador, o interpretador converte a linguagem máquina num executável, ficando assim o programa pronto para uso.

c)
Virtual machine - uma máquina virtual é a “virtualização” de um sistema computacional, isto é, é possível utilizar um ambiente computacional não físico dentro de um hardware já existente.

1.2

Este programa escreve os números até 3 em binário ———————————>

```c
#include <stdio.h>
void f (int n) {
  while (n>0) {
    if (n%2 == 0) putchar ('0');
    else putchar ('1');
    n = n/2;
  }
  putchar ('\n');
}
int main() {
    int i;
        for (i=0;(i<4);i++)
            f (i);
  return 0;
}
```

Para verificar as diferenças entre os ficheiros .s , mudei o tamanho do ciclo for.
Verificamos que, de facto, há diferenças nas instruções do código máquina (linha 61).

Para i<4:

```
51  ## %bb.0:
52      pushq   %rbp
53      .cfi_def_cfa_offset 16
54      .cfi_offset %rbp, -16
55      movq    %rsp, %rbp
56      .cfi_def_cfa_register %rbp
57      subq    $16, %rsp
58      movl    $0, -4(%rbp)
59      movl    $0, -8(%rbp)
60  LBB1_1:                                 ## =>This Inner Loop Header: Depth=1
61      cmpl    $3, -8(%rbp)
62      jge LBB1_4
63  ## %bb.2:                               ##   in Loop: Header=BB1_1 Depth=1
64      movl    -8(%rbp), %edi
65      callq   _f
66  ## %bb.3:                               ##   in Loop: Header=BB1_1 Depth=1
```

Para i<3:

```
51  ## %bb.0:
52      pushq   %rbp
53      .cfi_def_cfa_offset 16
54      .cfi_offset %rbp, -16
55      movq    %rsp, %rbp
56      .cfi_def_cfa_register %rbp
57      subq    $16, %rsp
58      movl    $0, -4(%rbp)
59      movl    $0, -8(%rbp)
60  LBB1_1:                                 ## =>This Inner Loop Header: Depth=1
61      cmpl    $4, -8(%rbp)
62      jge LBB1_4
63  ## %bb.2:                               ##   in Loop: Header=BB1_1 Depth=1
64      movl    -8(%rbp), %edi
65      callq   _f
66  ## %bb.3:                               ##   in Loop: Header=BB1_1 Depth=1
```

1.3

1.4

Em sistemas little-endian, os valores são armazenados do menos significativo para o mais significativo, isto é, da direita para a esquerda, pelo que, o valor de 32 bits lido em 4365 seria:
0000 1111 0000 0000 1111 1111 0001 0010
0 15 0 0 15 15 1 2

4362	0100 0011	4 3
4363	0111 0000	7 0
4364	0000 0011	0 3
→ 4365	0001 0010	1 2
4366	1111 1111	15 15
4367	0000 0000	0 0
4368	0000 1111	0 15

1.5
Se, acidentalmente, o programa modificar os valores armazenados na memória associados a uma instrução, esta pode passar a ser uma instrução que realiza algo diferente do inicialmente pretendido alterando assim o valor final do programa, ou pode até mesmo tornar-se numa instrução não executável assim como o programa.

1.6

	Tempo de produção		**Conversão**
Etapas	1 peça	1000 peças	Horas/1000peças
Preparação	20 s	20 000 s	5:33,20 s
Montagem	30 s	30 000 s	8:20 m
Testagem	35 s	35 000 s	9:43,20 s
Embalegem	35 s	35 000 s	9:43,20 s
Tempo total 1 pessoa:	120 s	120 000 s	1 dia 9:20 m
Tempo total 4 pessoas:	120 s	35 085 s	9:44,45 s

a) Uma pessoa levaria 1 dia, 9 horas e 20 minutos para produzir 1000 peças

b) 4 pessoas levariam 9 horas, 44 minutos e 45 segundos para produzirem 1000 peças.
Obti este resultado supondo que na 1 peça, o responsável pela embalagem teria que esperar pelos 3 anteriores até começar a trabalhar (um total de 85s), só depois todos estariam todos a trabalhar em simultâneo e ninguém teria que esperar mais pelo anterior.