Módulo nº 2

Assembly e ISA: IA32 vs. MIPS32

1. Introdução

Pretende-se com esta aula prática que os alunos identifiquem algumas diferenças fundamentais entre os níveis do *assembly* e do código-máquina de 2 arquitecturas com filosofias diferentes: IA32 e MIPS32.

Para atingir este objectivo o aluno deve desenvolver um pequeno programa em C e, usando as ferramentas `gcc`, `objdump`, `mips-gcc` e `mips-objdump`, acompanhar e visualizar as várias fases do processo de construção de um programa.

2. Linguagem de alto nível

Escreva em C, usando o editor de texto que considerar mais adequado, o seguinte programa:

prog.c

```
int accum=0;

int main ()
{
      int i;

      for (i=0 ; i<10 ; i++)
            if (maior(i, 5)) accum += i;
}

int maior (int p, int t)
{
      return (p>t ? 1 : 0);
}
```

Crie 2 subdirectorias (IA32 e MIPS32) e copie este ficheiro para cada uma delas.

3. Compilação IA32

Na directoria IA32 compile o programa `prog.c` usando o comando

`gcc -O2 -S prog.c`

Questão 1 – Identifique o mecanismo utilizado para passar parâmetros da função `main()` para a função `maior()`.

Questão 2 – Sabendo que a instrução **`call`** coloca o endereço de retorno (endereço da instrução a executar quando a função invocada terminar) na *stack*, qual será a instrução que o utiliza?

Questão 3 – A função `maior()` devolve o seu resultado para ser utilizado pela função `main()`. Como é que este valor é devolvido?

Questão 4 – Quantas instruções tem a função `main()`? E a função `maior()`?

Questão 5 – Quantas instruções são executadas no total, tendo em conta que o ciclo é iterado 10 vezes?

Questão 6 – Quantos acessos a dados em memória são feitos durante a execução do programa, tendo em conta que:

- **`call`** coloca o endereço de retorno na *stack*;
- **`ret`** lê o endereço de retorno da *stack*;
- **`leave`** lê o valor de **`%ebp`** da *stack*.

4. Compilação MIPS32

Na directoria MIPS32 compile o programa `prog.c` usando o comando

`mips-gcc –O2 –mrnames –S prog.c`

`NOTA 1: -mrnames` indica ao compilador que deve usar os nomes dos registos (**`$a0, $v0, $sp`**, etc.) em vez dos respectivos números.

`NOTA 2:` as instruções antecedidas pela directiva **`.set noreorder`** e agrupadas aos pares são ambas executadas pelo processador, apesar de a primeira ser sempre um salto. Semanticamente as instruções devem ser imaginadas como sendo executadas pela ordem inversa àquela com que aparecem no programa.

Questão 7 – Identifique o mecanismo utilizado para passar parâmetros da função `main()` para a função `maior()`.

Questão 8 –Onde é guardado o endereço de retorno das funções invocadas com a instrução **`jal`**?

Questão 9 – A função `maior()` devolve o seu resultado para ser utilizado pela função `main()`. Como é que este valor é devolvido?

Questão 10 – Não incluindo as instruções indicadas por **`#nop`**, quantas instruções tem a função `main()`? E a função `maior()`?

Questão 11 – Quantas instruções são executadas no total, tendo em conta que o ciclo é iterado 10 vezes?

Questão 12 – Quantos acessos a dados em memória são feitos durante a execução do programa, tendo em conta que no MIPS esta só é acedida pelas instruções **`lw`** e **`sw`**?

Questão 13 – Sabendo que o MIPS32 disponibiliza 32 registos e o IA32 disponibiliza 8 registos, e tendo em conta as respostas dadas às questões anteriores, o que pode concluir sobre o efeito que o número de registos disponibilizados tem sobre o desempenho de um programa?

5. Montagem (Assembler) IA32

Na directoria IA32 gere o código-máquina usando o comando

`gcc –O2 –c prog.s`

Visualize o ficheiro objecto usando os comandos

`objdump –d prog.o > prog.dump`

`vi prog.dump`

Questão 14 – O tamanho das instruções na arquitectura IA32 é variável. Qual a gama de tamanhos presente neste programa?

Questão 15 – Quantos bytes ocupa este programa?

Questão 16 – Parece-lhe que a referência à variável `accum` já está resolvida?

Questão 17 – Como estão codificados os destinos dos saltos **`je`** e **`jle`** e quais as instruções destino destes saltos?

Questão 18 – Parece-lhe que o destino do **`call`** já está resolvido? Qual o valor apropriado para este deslocamento?

6. Montagem (Assembler) MIPS32

Na directoria MIPS32 gere o código-máquina usando o comando

```
mips-gcc –O2 –c prog.c
```

NOTA: Não é possível usar o conteúdo do ficheiro **`prog.s`** porque o *assembler* **`mips-as`** não reconhece o nome dos registos. Se **`–mrnames`** não tivesse sido usado, então o **`mips-as`** poderia processar este ficheiro.

Visualize o ficheiro objecto usando os comandos

```
mips-objdump –d prog.o > prog.dump
vi prog.dump
```

Questão 19 – Quantos *bytes* ocupa cada instrução?

Questão 20 – Quantos bytes ocupa este programa?

Questão 21 – Parece-lhe que a referência à variável `accum` já está resolvida?

Questão 22 – Como estão codificados os saltos **`beqz`** e **`bnez`** ,sabendo que os 2 últimos *bytes* indicam o deslocamento e são armazenados em formato *big-endian*?

Questão 23 – Parece-lhe que o destino do **`jal`** já está resolvido? Qual o valor apropriado para este deslocamento?