Architecture des ordinateurs

L3 Informatique 2013-2014

TD 12 - Révisions

Dans tout le sujet, on utilise la syntaxe INTEL pour l'assembleur 80x86 (c'est à dire celle utilisée par NASM). Les réponses utiliseront la même syntaxe.

Exercice 1. On travaille sur une architecture disposant d'une mémoire RAM de 4 Go et de mémoire cache de niveau 2 (4 Mo), 1 (256 Ko) et 0 (32 Ko). On considérera que chacune de ces mémoires utilise des tailles de bloc (ligne) de 64 octets et une stratégie de remplacement de bloc consistant à éliminer le bloc accédé le moins récemment.

On réserve un espace mémoire de 1 000 000 mots machine (32 bits) que l'on interprête comme une image 1000×1000 : les mots 0 à 999 forment la ligne du haut (ordonnés de gauche à droite), les mots 1000 à 1999 forment la deuxième ligne, etc... On appelle chaque case du tableau un *pixel*.

(a) On effectue un traitement de l'image consistant à remplacer la valeur de chaque pixel par la moyenne des quatre pixels voisins (ligne ou colonne ± 1). Donner l'ordre de grandeur du nombre de cache miss sur chacun des niveaux 0, 1 et 2 (on négligera les phénomènes de bord de tableau).

(b) Même question sauf que cette fois on remplace la valeur de chaque pixel par la moyenne des pixels de la même ligne et de ceux de la même colonne.

Exercice 2. Proposer une traduction en assembleur 80x86 des séquences d'instruction C suivantes :

```
int a,b;
...
if (a<b)
    c=a;
else
    c=b;
```

```
int i,j;
j=0;
for (i=0;i<10;i++)
    j = j+i;
```

Exercice 3. On considère une architecture 80x86 32 bits et on suppose que le pointeur de pile sp vaut 1000. Indiquer la valeur du pointeur de pile et le contenu de la pile (connu) après chacune des instructions du code suivant :

```
            push ebx
            call addition
            pop ebx
            ret
addition:   and ebx,0xF0F0F0F0
            add eax,ebx
            ret
```

Exercice 4. Les deux codes ci-dessous représentent un code C et le résultat de sa compilation.

```
int callee(int, int, int);

int caller(void)
{
    int ret;
    ret = callee(1, 2, 3);
    ret += 5;
    return ret;
}
```

```
caller:
        push ebp
        mov ebp, esp
        push 3
        push 2
        push 1
        call callee
        add esp, 12
        add eax, 5
        pop ebp
        ret
```

(a) Que peut-on dire de la convention adoptée par ce système/compilateur concernant la transmission des paramètres lors de l'appel d'une fonction ?

(b) Proposer un code C à compiler et dont le désassemblage permettrait de confirmer que l'observation précédente est correcte. Indiquer ce à quoi vous feriez particulièrement attention dans le code compilé.

Exercice 5. On considère le code suivant s'exécutant dans le pipeline à 5 niveaux du 80x486 (lecture, décodage, décodage complémentaire, exécution, écriture).

```
1. mov ebx,[edi]
2. add edi,4
3. mov eax,[edi]
4. add edi,4
5. add ebx,eax
6. mov ecx,ebx
7. add ebx,eax
8. mov edx,ebx
```

(a) Combien d'étape de traitement faut-il au pipeline du CPU pour exécuter ce code ?

(b) Y-a t'il des temps d'attentes ("bulles") ? Si oui, combien et quelles en sont les causes ?

(c) À supposer que la réponse à la question (b) soit oui, proposer un code réalisant la même tâche et

Exercice 6. La table suivante donne les codes en langage machine de quelques instructions assembleur :

instruction	code
mov ebx,eax	0x89C3
xor ebx,ebx	0x31DB
add ebx,eax	0x01C3
add eax,eax	0x01C0
ret	0xC3

(a) Écrire un programme assembleur, à base d'instructions de la table ci-dessus, qui multiplie le registre `eax` par 20.

(b) Traduire le code de la question (a) en langage machine.

(c) Écrire un programme assembleur qui génère un code, à base d'instructions de la table ci-dessus, qui multiplie le registre `eax` par une constante c 8 bits que l'on supposera donnée dans le registre `ecx`.