université angers
FACULTÉ DES SCIENCES
Unité de formation et de recherche

Département Informatique
L3 Inforamtique
Jean-Michel Richer
UE Architecture des Ordinateurs
2014/2015

jean-michel.richer@univ-angers.fr

Examen (2h)

Exercice 1 - (4 pts) - Questions de cours

On considère une chaine de production (CP1) composée par un opérateur qui doit réaliser 10 opérations de 1 minute chacune afin d'assembler les différentes parties d'un iPhone.

1. combien de produits obtiendra t-on après 4h de travail avec CP1 ?
2. CP2 représente la même chaine de production mais organisée sous forme de pipeline. Combien de produits fabriquera t-on après 4h de travail ?
3. combien de chaînes de type CP1 faut-il pour concurrencer une chaîne de type CP2 ?

Exercice 2 - (6 pts) - Représentation

Les représentations hexadécimales des nombres flottants suivants sont :

$1.5 =$	3F.C0.00.00		$6.0 =$	40.C0.00.00
$3.0 =$	40.40.00.00		$12.0 =$	41.40.00.00

1. en se basant sur cette table quelle est la représentation hexadécimale de 24.0 ?
2. écrire le code assembleur d'un sous-programme qui multiplie par 2 un nombre flottant passé en paramètre dans le registre EAX, le résultat sera placé dans EAX. On considère qu'il n'y aura pas de dépassement de capacité.
3. faire de même pour la fonction `neg` qui changee le signe du nombre placé dans EAX

Exercice 3 - (10 pts) - Codage assembleur

Soit la fonction suivante ou n est un multiple de 4 et x, y et z sont des adresses multiples de 16 :

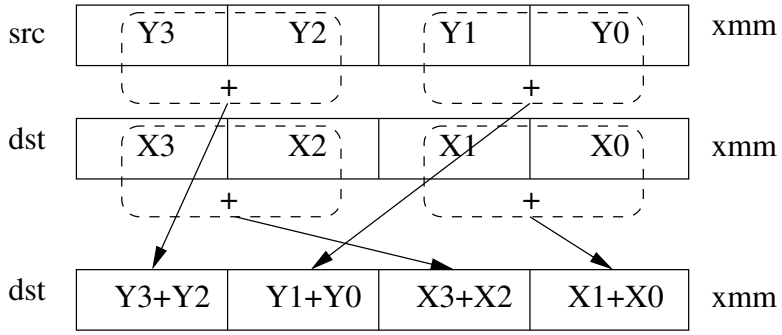
```
float f(float x[], float y[], float z[], int n) {
  int i; float sum = 0.0;
  for (i = 0; i < n; ++i) {
    sum += x[i] / (y[i] + z[i]);
    x[i] = sqrt(y[i] * z[i]);
  }
  return sum;
}
```

1. coder f en assembleur x86 32 bits
2. donner une version avec un dépliage par 4

3. donner une version vectorisée de f en utilisant les instructions SSE

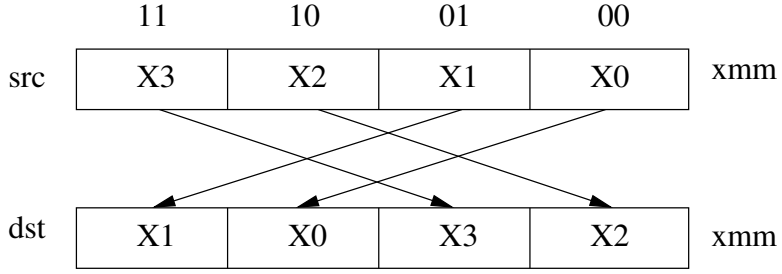
On rappelle que les instructions SSE liées aux réels sont :

— `movaps xmm0, [adrx16]` : charge dans `xmm0` les données situées à l'adresse `adrx16`
— `movaps [adrx16], xmm0` : stocke à l'adresse `adrx16` les données de `xmm0`
— `addps xmm0, xmm1` : additionne 4 flottants en parallèle
— `mulps xmm0, xmm1` : multiplication de 4 flottants en parallèle
— `divps xmm0, xmm1` : division de 4 flottants en parallèle
— `sqrtps xmm0, xmm1` : calcul de la racine carrée de 4 flottants en parallèle
— `movss [adr], xmm0` : copie les 32 bits situés à `adr` dans les 32 premiers bits de `xmm0`
— `movss xmm0, [adr]` : copie les 32 premiers bits de `xmm0` à l'adresse `adr`
— `haddps xmm0, xmm1` : addition *horizontale* sur des réels



haddps xmm1,xmm2

— `shufps xmm0, xmm1, imm8` : copie les valeurs sur 32 bits du registre source dans le registre destination en indiquant lesquels choisir



shufps xmm0,xmm1,01001110b