université angers
FACULTÉ DES SCIENCES
Unité de formation et de recherche

Département Informatique
L3 Informatique
Jean-Michel Richer
UE Architecture des Ordinateurs
2020/2021

jean-michel.richer@univ-angers.fr

Examen - 1h30

Il sera tenu compte dans la notation de la lisibilité de la copie, du style, de l'orthographe ainsi que des explications fournies.

Exercice 1 - (3 pts) - Algèbre de Boole

1. Simplifier la fonction booléenne de 4 variables $f(A, B, C, D) = (1{-}3, 6, 8{-}11, 14, 15)$ en utilisant un tableau de Karnaugh et donner sa formule simplifiée
2. Simplifier l'expression suivante en utilisant les formules algébriques :

$$\overline{(A + CD)}.\overline{(\overline{AB} + \overline{C})}$$

Exercice 2 - (6 pts) - Nombre de Mersenne premier

Un nombre de Mersenne premier est un nombre de la forme $2^n - 1$ qui est premier. Donnez le code assembleur 32 bits du programme C suivant qui cherche les nombres de Mersenne premiers.

```
bool is_prime(int n) {
    if (n <= 1) return false;
    if (n <= 3) return true;
    if ((n % 2) == 0) return false;
    for (int i = 3; i <= static_cast<int>(sqrt(n)); i += 2) {
        if ((n % i) == 0) return false;
    }
    return true;
}

int main() {
    int n;
    for (n = 1; n <= 30; ++n) {
        int v = (1 << n) - 1 ;
        printf("%d %d %d\n", n, v, is_prime(v));
    }
    return 0;
}
```

On rappelle qu'il existe une instruction `fist/fistp [mem]` qui convertit la valeur dans `st0` en un entier.

Exercice 3 - (7 pts) - Codage assembleur

```
float prod_sum(float *x, float *y, float *z, int size, float a) {
    float sum = 0.0;

    for (int i = 0; i < size; ++i) {
      z[i] = x[i] * y[i] / a;
      sum += z[i];
    }
    return sum;
}
```

1. donnez le code assembleur x86 32 bits de la fonction `prod_sum` sans vectorisation. On indiquera ce que contiendront les registres.
2. donnez le code C/C++ avec un dépliage par 4
3. donnez une version assembleur vectorisée (SSE) de la fonction en utilisant les instructions `movaps`, `mulps`, `addps`, `divps` et `haddps`

Note : on considère que les vecteurs `x` ,`y` et `z` sont alignés sur des adresses multiples de 16.