Examen M1 informatique

J.-M Friedt, 5 avril 2018

Connexion internet interdite, téléphones portables interdits, communications interdites, réflexion autorisée.
Supports de cours à `172.20.133.70`

Nous considérons le logiciel qui sera embarqué dans une sonde pour aller sur Mars. Les diverses étapes de vol consistent à annoncer le décollage en allumant la LED appropriée, puis à calculer la circonférence de la Terre. Une fois le vol amorcé, nous devrons séquencer plusieurs tâches. Finalement lors de l'arrivée à destination, plusieurs calculs sont effectués en parallèle et leurs résultats transmis simultanément.

1 Programmation Atmega32U4 (2 points)

1. **Une LED est connectée sur un port de l'Atmega32U4. En consultant le schéma de la carte et la documentation technique du processeur, tous deux disponibles dans l'archive, faire clignoter à 1 Hz la LED.**

On fournit comme exemple de programme fonctionnel le contenu de l'archive `atmega32`.

2 Programmation bas niveau sur STM32 (2 points/question)

Nous fournissons une archive d'un programme fonctionnel sur STM32 qui affiche un message. En partant de l'archive `baremetal_stm32`, compilez le programme et exécutez le sous `qemu` par
`qemu-system-arm -M stm32-p103 -serial stdio -serial stdio -kernel main.bin`
avec `qemu-system-arm` disponible dans `/home/jmfriedt/qemu_stm32/arm-softmmu` [1].

2. **Comment sélectionner les 4 bits de poids faible d'une variable ?**
3. **Écrire une fonction qui affiche sur le port de communication, en s'appuyant sur la fonction d'affichage d'une chaîne de caractères, le contenu de la variable `rayon` en hexadécimal. Il peut être judicieux de développer ce code source dans un fichier séparé afin de le réutiliser dans la section suivante. Afficher le rayon de la Terre en hexadécimal depuis le STM32 émulé par `qemu` ?**
4. **Connaissant le rayon de la Terre, calculer sa circonférence en kilomètres, en suivant les consignes de programmation vues en cours, avec deux décimales exactes. Afficher ce résultat en décimal au moyen de `qemu`. Il est envisageable qu'il faille modifier la fonction d'affichage pour arriver à ce but.**

3 Programmation sur STM32 sous FreeRTOS (2 points/question)

Nous nous intéressons maintenant à séquencer `N` itérations d'une même fonction pour afficher le résultat d'un calcul sous FreeRTOS. On s'inspirera pour cela de l'archive `FreeRTOS`. On commencera par démontrer la compilation et l'exécution du programme d'exemple dans `qemu`.

5. **Lancer $N = 5$ tâches identiques et observer leur comportement. Que constatez vous comme dysfonctionnement ?**
6. **Corriger ce dysfonctionnement en exploitant le mécanisme approprié.**
7. **Passer *trois arguments*, au lancement de chaque tâche, correspondant à 1000, 1333 et 2000 fois l'indice de la tâche, et compléter l'affichage de chaque tâche par ces valeurs.**

4 Questions de cours (1 point/question)

8. **Quel mécanisme intrinsèque à la majorité des processeurs supportant un système d'exploitation interfère avec l'accès aux ressources matérielles ?**
9. **En quel emplacement de la RAM se trouve classiquement la pile ?**
10. **Quel est la durée de vie d'une variable placée sur la pile ?**
11. **En quel emplacement de la RAM se trouve classiquement le tas ?**
12. **Quel est la durée de vie d'une variable placée sur le tas ?**
13. **Que vaut 0x234×4 ? (dans la base de votre choix). Justifier comment ce résultat est obtenu.**

1. Noter que `qemu` prend en argument le fichier `bin` issu de la conversion de l'exécutable au format ELF par `objcopy`

Corrections

1. D13 = PC7 donc penser à placer cette broche en sortie (`DDRC|=0x80;`) et manipuler ce bit dans `PORTC`.
2. `variable & 0x0f`
3. la sortie du programme fourni en exemple lors de son exécution est

   ```
   Hello World
   Hello World
   ...
   ```

   6400 en décimal s'écrit 0x1900, tel que nous le constatons avec

```
#include "common.h"
void affiche(short s)
{int k;char c;
 for (k=3;k>=0;k--)
   {c=(s>>k*4)&0x0f;
    if (c<10) uart_putc('0'+c); else uart_putc('A'+c-10);
   }
}

int main(void){
 short rayon=6400;
 Usart1_Init(); // inits clock as well
 while(1){affiche(rayon);uart_putc('\n');}
 return 0;
}
```

4. Le calcul d'incertiture est le suivant : sachant que nous désirons obtenir la circonférence C avec deux décimales exactes, l'incertitude $d\pi$ sur π est de

$$C = 2\pi R \Rightarrow dC = 2Rd\pi \Leftrightarrow d\pi = \frac{dC}{2R} = \frac{10^{-2}}{2 \times 6400} = 8 \times 10^{-7} \simeq 10^{-6}$$

   donc il faut 6 décimales de π. Les deux fractions rationnelles classiques – que l'on retrouve par la fonction `rats()` de GNU/Octave – de π sont $\pi \simeq 355/113$ et $\pi \simeq 22/7$. Dans les deux cas, la multiplication du numérateur par 6400 fait dépasser la capacité de stockage d'un `short` : il faut donc passer par une variable intermédiaire en long. Cependant, 22/7 ne présente pas suffisamment de décimales exactes et le résultat est erroné. Seule la fraciton 355/113 permet d'atteindre le résultat. Mis à part cette subtilité, l'affichage se réduit à

```
#include "common.h"

void affiche(long val)
{int k=10000000;
 while (k>10) {uart_putc('0'+val/k);val-=(val/k)*k;k/=10;}
 uart_putc('.'); // decimales
 while (k>=1) {uart_putc('0'+val/k);val-=(val/k)*k;k/=10;}

}

int main(void){
 short rayon=6400;
 long circonference;
 Usart1_Init(); // inits clock as well
 circonference=((long)rayon*35500*2)/113;
 while(1){affiche(circonference);uart_putc('\n');}
 return 0;
}
```

   La solution de multiplier par 3141592 puis de diviser par 1000000 n'est pas viable car ce numérateur multiplié par 6400 dépasse 2^{32}.
5. la sortie du programme lors de son exécution est

   ```
   BonjourBonjBonjour l la valeuour la vala valeur r de la meur de lade la mesesure est mesure eure est  st
   BonjourBonjour lBonjour l la valeua valeur da valeur r de la me la mesude la meesure estre est su re est
   Bonjour laBonjour lBonjour l valeur da valeur a valeur e la mesde la mesude la mesure est rure est e est
   ```

6. Protéger l'affichage par un mutex
7. Définir une structure donc les éléments sont les arguments passés en paramètre.

   ```
   struct troisvaleurs {int i1;int i2;int i3;};
   ```

Penser à définir cette structure en `static` dans le main :

`static struct troisvaleurs v[NBTASK];`

et finalement caster le passage d'argument de `void*` en `struct*`

```
struct troisvaleurs *argument;
argument=(struct troisvaleurs*)dummy;
```

```
#include "FreeRTOS.h"
#include "task.h"
#include "semphr.h"
#include "common.h"

#define NBTASK 3
struct troisvaleurs {int i1;int i2;int i3;};
xSemaphoreHandle xMutex;

void vPrintUart(void* dummy)
{struct troisvaleurs *argument;
 argument=(struct troisvaleurs*)dummy;
 while(1){
          xSemaphoreTake( xMutex, portMAX_DELAY );
          uart_puts("Bonjour la valeur de la mesure est \0");
          affiche(argument->i1,c); uart_puts(c); uart_putc(' ');
          affiche(argument->i2,c); uart_puts(c); uart_putc(' ');
          affiche(argument->i3,c); uart_puts(c); uart_putc(' ');
          xSemaphoreGive( xMutex );
          vTaskDelay(100);
   }
}

int main(void){
 static struct troisvaleurs v[NBTASK];
 int k;
 Usart1_Init(); // inits clock as well
 xMutex=xSemaphoreCreateMutex();
 for (k=0;k<NBTASK;k++)
    {v[k].i1=1000*k;
     v[k].i2=1333*k;
     v[k].i3=2000*k;
     xTaskCreate( vPrintUart, (signed char*) "Uart", 128,(void*)&v[k],3,NULL );
    }
 vTaskStartScheduler();
 return 0;
}
```

8. Memory Management Unit – MMU
9. à l'adresse la plus élevée : le pointeur de pile est *décrémenté* quand on empile une variable
10. locale à la durée d'exécution de la fonction
11. à l'adresse la plus basse de la RAM
12. la durée d'exécution du programme
13. 0b001000110100 << 2 = 0b100011010000 =0x8D0