

Microprocesseurs 1 & 2: Travail écrit no 3.

Nom :

Prénom :

Classe : I/2

Date : 22.04.2013

Problème n° 1 (interfaçage C - assembleur)

a) Codez en assembleur la fonction « f1 » ci-dessous :

```
typedef int (*op_t) (int p1);
int f1 (int a1, int a2, op_t op1, int a3) {return a1 + a2 + op1(a3);}
```

b) Dessinez l'état de la pile pour la fonction « f2 » ci-dessous juste avant l'appel de la fonction « op2 ». Le compilateur utilise le « frame pointer » (fp) pour accéder les paramètres.

Indiquez sur le graphique la position du « fp » et du « sp ».

```
struct s {int s1; int s2; int s3;};
extern int op2(int a1, const struct s* a2);
int f2 (int p[4], int p5, int p6, int p7, const struct s* p8)
{return p[0] + p[1] + p[2] + p[3] + p5 + p6 + op2(p7, p8);}
```

c) Citez les techniques et méthodes utilisées pour le passage d'arguments/paramètres à des fonctions.

Microprocesseurs 1 & 2: Travail écrit no 3.

Problème n° 2 (Interruptions)

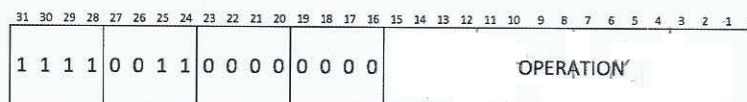
- a) Décrivez succinctement le mécanisme implémenté par le processeur ARM pour trouver et appeler la routine de service d'interruption (ISR) lors de la levée d'une interruption (IRQ/FIQ/SWI) ou d'une exception (undef/data abort/prefetch abort)
- b) Décrivez succinctement le système d'interruption de l'i.MX27 du point de vue HW et indiquez comment attacher un périphérique externe au système d'interruption de l'i.MX27 (côté HW)
- c) Décrivez succinctement la **commutation de contexte** et la **latence** d'interruption

Microprocesseurs 1 & 2: Travail écrit no 3.

Problème n° 3 (Interruptions)

Dans le jeu d'instructions des processeurs ARMv5, codé sur 32 bits, il existe un certain nombre d'instructions qui ne sont pas définies. Ces instructions, si elles sont exécutées, génèrent l'exception « undef ».

Un programmeur décide d'utiliser ces instructions pour créer un nouveau jeu d'opérations. La figure ci-dessous représente le format du jeu d'opérations supplémentaires. Les bits 31 à 16 permettent d'identifier le nouveau jeu d'opérations. Le numéro de l'opération à exécuter est codé sur les bits 15 à 0. Les registres R1, R2 et R3 permettent de passer trois arguments (a1, a2 et a3) à l'opération. Le résultat de celle-ci est finalement retourné dans le registre R0.



Pour rappel, l'adresse de retour sauvee par le processeur lors de l'exécution de cette instruction n'a pas d'offset / de décalage.

- a) Implémentez en assembleur la routine d'interruption permettant de traiter le nouveau jeu d'opérations en appelant la routine « int process_ops (int operation, int a1, int a2, int a3) ». Si le contenu des bits 31 à 16 ne correspond pas au pattern ci-dessus, la valeur -1 sera retournée.

- b) Citez les étapes/la marche à suivre pour installer cette routine d'interruption sur le microprocesseur

Microprocesseurs 1 & 2: Travail écrit no 3.

Problème no 4 (entrées/sorties)

- a) Décrivez succinctement le traitement des entrées/sorties par interruption. Utilisez un schéma de principe pour soutenir la description.
- b) Calculez la latence maximale autorisée côté application pour la réception de paquets d'un contrôleur Ethernet à 1Gbps full-duplex. Le pilote gérant le contrôleur Ethernet est implémenté en mode d'interruption. Il dispose d'un tampon circulaire de 500 entrées avec une taille de 1500 bytes par entrée. Côté réseau, les paquets de 250 octets sont émis à intervalle de 10 us.
- c) Implémentez la routine « void putchar(char c) » permettant d'émettre un caractère sur une interface série au travers du contrôleur ci-dessous. L'émission se fera par scrutation.

```
#define STATUS_RXRD (1<<0) /* receiver ready: character has been received */
#define STATUS_TXRD (1<<1) /* transmitter ready: character could be sent */
struct serial_ctrl {
    uint8_t status; /* registre de statut voir bits ci-dessus */
    uint8_t control; /* registre de contrôle du périphérique */
    uint8_t tx; /* registre pour l'émission des données */
    uint8_t rx; /* registre contenant les données reçues */
};

static volatile struct serial_ctrl* serial = (struct serial_ctrl)0x10008000;
```

Microprocesseurs 1 & 2: Travail écrit no 3.

Problème n° 5 (système temps réel)

a) Citez les composantes principales d'un noyau temps réel

b) Citez, à l'aide d'une machine d'états, les états d'un thread et indiquez leur fonction.

c) Sur la base des structures et variables ci-dessous, implémentez la fonction « void sema_wait (int id) », permettant de prendre le sémaphore.

```
struct tcb { /*...*/ enum states state; struct tcb* chain; /*...*/ };  
struct semaphore { long count; bool is_used; struct tcb* chain; };  
struct tcb* running_thread;  
struct semaphore sema[200];  
extern rtos_reschedule();
```