



Systemes Embarqués 1 & 2: Travail écrit no 3.

Nom :

Prénom :

Classe : T-2/I-2

Date : 20.04.2015

Problème n° 1 (interfaçage C - assembleur)

- a) Codez en assembleur la fonction « bar » ci-dessous. Note : les « long » ont 32 bits.

```
typedef long (*foo_t) (long a1, long a2);  
long bar (long p1, foo_t foo, long p2, long p3, long p4, long p5)  
{return p3 + p2 + foo (p5, p1) + p4;}
```

- b) Le graphique ci-dessous représente l'état du processeur (registres et pile sur 32 bits) à l'entrée de la fonction « f2 » (aucune instruction de « f2 » n'a encore été exécutée).

```
int f2 (int a1, int a2, struct S* a3) {return a2 + f3(a1, a3);}
```

Indiquez la valeur des paramètres a1 à a3.

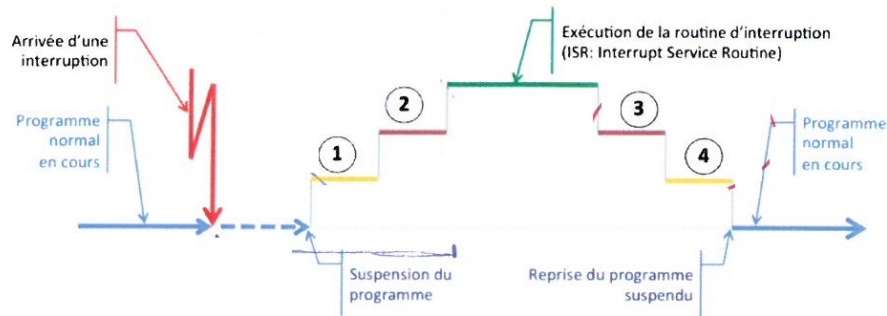
low address	100	R0	10
	414		1020
	153		0
	SP (à l'entrée de f2) → 0		45
	567		18
	898		90
	910		33
high address	1114	R6	

- c) Implémentez en assembleur le passage par valeur de la variable « int var ; » dans le registre R0

Systèmes Embarqués 1 & 2: Travail écrit no 3.

Problème n° 2 (Interruptions)

La figure ci-dessous représente les différentes transitions que le microprocesseur et son logiciel effectuent lorsque une interruption est soulevée avant de pouvoir exécuter la routine d'interruption (ISR).

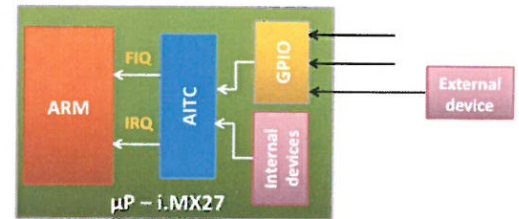


- Décrivez succinctement les opérations effectuées lors des transitions 1 à 4. Précisez si celles-ci sont effectuées par le microprocesseur (hw) ou par le logiciel (sw).
- Implémentez les opérations assembleur que le microprocesseur devra exécuter pour sauvegarder et restaurer l'état du microprocesseur lorsqu'une interruption matérielle (IRQ ou FIQ) est levée. Indiquez le numéro de la transition où s'effectuent ces opérations
- Représentez sur le graphique ci-dessus la commutation de contexte d'interruption.
- Représentez sur le graphique ci-dessus la latence.
- Donnez le terme technique de la variation du temps de latence et citez 2 exemples qui peuvent la faire varier fortement.

Systèmes Embarqués 1 & 2: Travail écrit no 3.

Problème n° 3 (Interruptions)

La figure ci-contre représente le système d'interruption du microprocesseur i.MX27.



- a) Décrivez succinctement le rôle des composants ci-dessous et indiquez la méthode utilisée par le composant pour identifier la source d'interruption.
- ARM :

 - AITC :

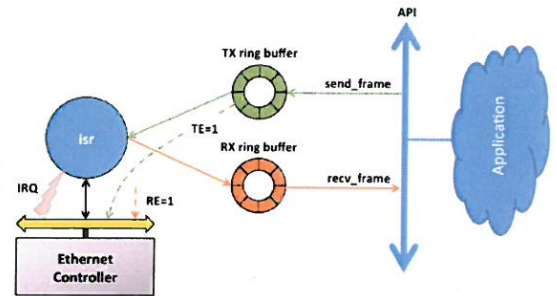
 - GPIO :
- b) Implémentez en assembleur la fonction « `void init_sp (int mode, void* sp);` » permettant d'initialiser les pointeurs de piles pour les différents modes du μP (registre SP).
init_sp :
- c) Indiquez le type de pile utilisée et comment celle-ci fonctionne.
- d) Sachant que « `var` » est une variable de type « `int` », décrivez pour quelle raison l'opération « `var++;` » n'est pas atomique et indiquez les instructions à effectuer pour rendre cette opération atomique.

Systèmes Embarqués 1 & 2: Travail écrit no 3.

Problème n° 4 (Entrées/Sorties)

La figure ci-contre représente le schéma de principe pour le traitement par interruption d'un contrôleur pour une interface Ethernet à 100Mbit/s full-duplex.

- a) Décrivez la fonction des tampons de réception et d'émission (RX ring buffer et TX ring buffer).



- b) Décrivez le rôle de la routine d'interruption (ISR).

- c) Implémentez la routine « void send_frame (const msg_t* frame) ; » permettant à l'application d'émettre des trames.

```
#define CTRL_TE (1<<1) // transmitter interrupt enabled
static volatile struct ethernet_ctrl {
    uint16_t ctrl; // control register
    /* ... other registers... */
} * ether = (struct ether_ctrl*)0x2001c000;
struct fifo {int in; int out, msg_t frames[512];} tx_fifo;

void send_frame (const msg_t* frame) {
```

- d) Pour le contrôleur Ethernet ci-dessus et le mode de traitement par interruption, dimensionnez le tampon de réception afin que la latence maximale autorisée côté application soit au minimum de 45 ms. Afin d'économiser la taille mémoire nécessaire, le tampon de réception sera formé de blocs de 200 bytes chacun. Si la taille du paquet dépasse la taille maximale d'un bloc, celui sera stocké sur plusieurs blocs. Côté réseau des paquets de 125 bytes et/ou 1250 bytes (framing compris) sont émis en burst de 5 ms à plein débit et suivi ensuite d'une pause de 10 ms.

Systèmes Embarqués 1 & 2: Travail écrit no 3.

Problème n° 5 (Systèmes temps-réel)

- a) Citez les composants principaux du noyau d'un système d'exploitation.
- b) Définissez la structure minimale du TCB (Thread Control Block)
- ```
struct tcb {
```
- c) Implémentez la fonction de transfert implémentant la commutation de contexte entre deux threads.
- ```
void transfer (struct tcb* former, uint32_t psr, struct tcb* new) ;  
transfer :
```
- d) Définissez la structure minimale d'un sémaphore.
- ```
struct semaphore {
```
- e) Implémentez la fonction « void sema\_signal(int id) ; » permettant de libérer le sémaphore.
- Quelques éléments de réalisation :
- ```
struct semaphore sema [200];  
struct tcb* sema_extract_thread (int sema_id); // extrait 1er tcb contenu dans la  
liste du sémaphore, retourne 0 si aucun tcb n'est dans la liste  
void sema_signal(int id) {
```