4.8

Systèmes Embarqués 1 & 2: Travail écrit no 4.

Nom : [redacted] **Prénom :** [redacted]

Classe : T-2/I-2
Date : 16.06.2017

Problème n° 1 (Entrées/Sorties)

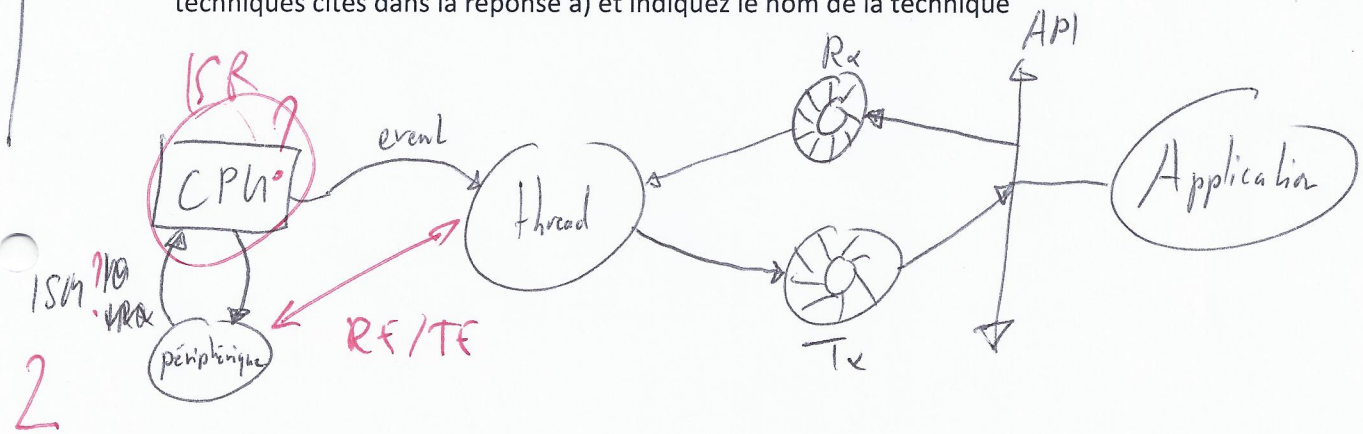
Une équipe de développement logiciel est chargée de concevoir un pilote pour une interface Ethernet à 100Mb/s full-duplex. Le cahier des charges stipule que l'application ne devra perdre aucun paquet de données et la latence maximale que le pilote doit pouvoir absorber est de 80 ms. Sur une période de 20 ms, l'interface va recevoir une rafale (burst) de paquets de 1000 bytes pendant 4 ms, puis une rafale de paquets de 250 bytes durant 6 ms.

a) Indiquez les différentes techniques que l'on peut mettre en oeuvre pour servir le contrôleur Ethernet afin de satisfaire au mieux le cahier des charges ci-dessus.

- Accès direct (pas de buffer) ← (ne convient pas car on doit accepter la latence)
- Accès périodique (buffer)
- Interruption ISR (buffer + ISR)
- Interruption avec traitement par thread (buffer + ISR + thread)

1 1/2

b) Décrivez à l'aide d'une figure le principe mis en œuvre pour servir le périphérique selon l'un des techniques cités dans la réponse a) et indiquez le nom de la technique



2

Quand une ISR arrive, le cpu fait un event sur le thread, il va le mettre en running et le thread va lire Rx et on écrire dans Tx (ce sont des buffers).

Systèmes Embarqués 1 & 2: Travail écrit no 4.

c) Dimensionnez le tampon de réception afin de satisfaire le cahier des charges ci-dessus. Dans le but d'économiser de l'espace mémoire, le tampon de réception sera formé de blocs de 500 bytes chacun. Si la taille du paquet reçu dépasse la taille maximale d'un bloc, celui sera stocké dans plusieurs blocs. Si le dernier bloc n'est pas complètement utilisé, la place restante ne sera pas utilisée pour un autre paquet.

100Mb/s latence = 80ms 4ms packet 1000 bytes + 6ms 250B
8000 b. 2000 b.

100Mb/s pendant 1ms = 100Kb

en 20ms
4ms = 400Kb = 400'000 Bits = 50 packets de 1000B = 100 blocs
6ms = 600Kb = 600'000 bits = 300 packets de 250B = 300 blocs

en 80ms => 400 blocs
400 blocs
> 1600 blocs ✓

Le tampon doit faire au minimum 1600 blocs

soit 800'000 Bytes = 800KB

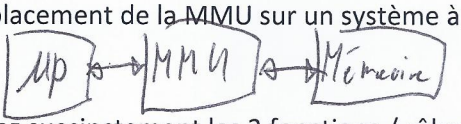
5

9

Systèmes Embarqués 1 & 2: Travail écrit no 4.

Problème n° 2 (MMU)

1. Indiquez l'emplacement de la MMU sur un système à µP

1 

Entre mémoire et µP

2. Citez et décrivez succinctement les 3 fonctions / rôles principaux de la MMU dans un système à µP

- Convertir adresse [illegible] virtuelle – réelle ✓
- On permet à chaque Process de croire avoir toute la mémoire
  → C'est une conséquence de point 1
- Augmente la sécurité en ne permettant pas aux process d'accéder directement à la mémoire.

1 1/2

3. Donnez la définition de l'abréviation TLB, décrivez la fonction de la TLB et indiquez son emplacement

Translation Lookaside Buffer, il est dans le MMU et c'est un cache pour une translation rapide entre adr virtuelle – réelle
Il regarde simultanément toutes les entrées dans son cache pour savoir si l'adresse demandée y figure

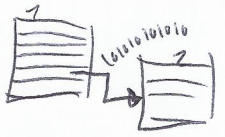
2

4. La MMU du processeur TI AM335x implémente une table à deux niveaux (1er niveau sur 12 bits et 2e niveau sur 8 bits). Décrivez succinctement la raison de cette implémentation sur 2 niveaux et décrivez à l'aide d'un exemple comment le µP procède pour accéder des données placées dans une page du 2e niveau.

adresse 1010 1010 1010 1111 1111 .....

0 2

On cherche dans la table de premier niveau la table à l'adresse 1010 1010 1010 (l'index)



Puis dans la table de deuxième niveau, on prend l'entrée 1111 1111 (la dernière).

5. Indiquez la fonction des sections / pages « fault ».

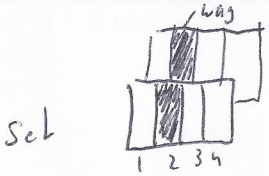
1/2 page fault = la page est lue / écrit une erreur.
↳ je n'arrive pas à lire

(7)

Systèmes Embarqués 1 & 2: Travail écrit no 4.

Problème n° 3 (Mémoire cache)

1. Décrivez succinctement l'architecture et le principe de fonctionnement d'une mémoire cache à n-voies associative.



2 1/2 Mix entre accès direct et complètement associatif. On choisit un Set selon l'accès direct et on choisit la way dans le set selon complètement associatif.

Offre un excellent compromis entre vitesse, ~~sécurité~~ prix.

2. Hormis les données, dans une ligne de la mémoire cache on trouve encore trois champs, soit le tag, le bit V et le bit D. Décrivez succinctement la fonction de ces trois champs

tag = l'adresse de la ligne (où elle est stockée)

1 1/2 V = valid

d = dirty (modifié par l'uP)

3. Citez et décrivez succinctement les deux principes qui sont à l'origine des mémoires caches. Pour chacun des principes, donnez un exemple.

Localité Spatial : Si on sauve une donnée à un endroit de la mémoire, il est très probable que l'on sauve qqch d'autre à la suite dans un courts laps de temps. Exemple = initialisation de variable : int loop = 10 ;
int test = 0 ;

3

Temporelle : Si on sauve qqch dans la mémoire, il est très probable qu'on va l'utiliser rapidement. Exemple : On init i et on l'utilise directement.
for(int i = 0 ; i < ... ; i++){

3

4. Quel principe est utilisé par la mémoire cache des instructions (i-cache) et donnez-en la raison

?

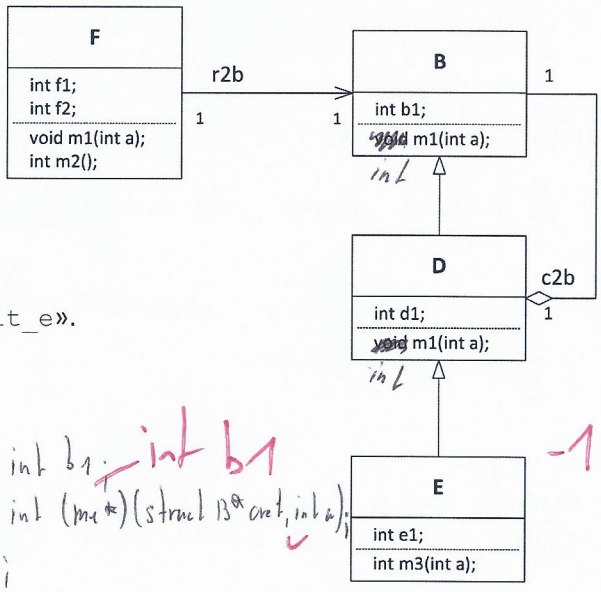
7

Systèmes Embarqués 1 & 2: Travail écrit no 4.

Problème n° 4 (programmation orientée-objet)

Pour le diagramme de classes ci-contre :

1. Déclarez les classes F, B, D et E en langage C orienté-objet.
   Remarque : la méthode «`m1`» de la classe D surcharge celle de la classe B.
2. Implémentez la fonction «`m1`» de la classe D de manière à ce qu'elle retourne la somme « `a + d1 + b1` »
   La macro « `container_of` » est à disposition.
3. Implémentez la méthode d'initialisation de la classe E «`init_e`».
   Remarque : la variable e1 doit être initialisée à 14;




```
struct E { int e1;
           int (m3*)(struct E* oret, int a);
           struct D d_ret; };

struct D { int d1;
           struct B b_ret;
           struct B* c2b; };

struct B { int b1;
           int (m1*)(struct B* oret, int a); };

struct F { int f1, f2;
           void (m1*)(struct F* oret, int a);
           int (m2*)(struct F* oret);
           struct B* r2b; };

int m1(struct D* oret, int a){
    struct B* b_base = container_of(oret, struct B, c2b);
    return a + oret->d1 + b_base->b1; }

void init_e(int arg){
    struct E ee={14};
```

7

Systèmes Embarqués 1 & 2: Travail écrit no 4.

Problème n° 5 (Toolchain + DMA)

1. Expliquez à l'aide d'un exemple simple le principe d'un Makefile

Permet de rennir les flag, commande etc... nécessaire à la compilation dans un fichier. Ensuite, on peut build avec un make qui appelle le makefile au lieu de faire chaque fois toute les comenandes.

1

2. Citez 2 ou 3 avantages des revues de code

- Permet de valider le fonctionement de méthode
- Permet d'avoir une vue externe sur le code, car celui qui fait la revue n'est pas la même personne que celle qui à codé.

2

3. Implémentez un test unitaire permettant de valider/vérifier le bon fonctionnement de la fonction ci-dessous (3 tests positifs et 3 tests négatifs).

```
/**
 * The C library function int tolower(int c) converts a given letter to
 * lowercase.
 *
 * @param c - This is the letter to be converted to lowercase.
 *
 * @return - This function returns lowercase equivalent to c, if such value
 *           exists, else c remains unchanged. The value is returned as an
 *           int value that can be implicitly casted to char.
*/
int tolower(int c);
```

(int)'A' (int)'a'

```
c_assert( tolower(A)==a) report error;
                (int)'b'
c_assert ( tolower( b)== b ) report error;
c_assert ( tolower (65) == (int)'a') report error;
c_assert ( to_lower (1000)== 1000) report error;
c_assert ( to_lower ((int)'!' == (int)'!') report error;
```

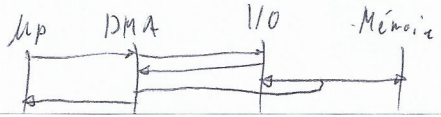
2

5

4. Donnez la définition de l'abréviation DMA et décrivez succinctement sa fonction dans un système à µP

Direct Memory Access | Permet de libérer le µP et de gérer l'accès à la mémoire par les périphériques à sa place.

2



1 = Configuration
2 = ready
3 = échanges
4 = fin donc