UFR IMA

Informatique & Mathématiques Appliquées

Grenoble 1

UNIVERSITE
JOSEPH FOURIER

Sciences, Technologie, Médecine

Institut des Sciences et Techniques de Grenoble – RICM-1

Lundi 6 Janvier 2003

EXAMEN D'ARCHITECTURE LOGICIELLE ET MATERIELLE

Durée 3 heures
Tous documents autorisés

Question 1: (4 points)

On s'intéresse ici à l'étude d'un codage appelé « décimal codé binaire » (DCB). Ce codage consiste pour un naturel à coder en base deux sur 4 bits chacun de ses chiffres décimaux.

Exemples :

l'entier 97 en décimal a pour code : 1001 0111 en DCB.

l'entier 15 en décimal a pour code : 0001 0101 en DCB.

Pour un chiffre décimal, les codes 1010 à 1111 sont des codes invalides.

A- Donnez la description à base de portes NAND et d'inverseurs d'un circuit comportant 32 entrées et délivrant une sortie qui vaut 1 si l'entrée sur 32 bits est un code DCB valide (0 sinon).

Exemple : en hexadécimal : Entrées= 0x11022095, sortie=1

Entrées= 0x12B77766, sortie=0

B- On s'intéresse à l'addition de deux nombres codés en DCB

Donnez le programme en langage d'assemblage ARM qui calcule la somme modulo 10^8 de deux naturels codés en DCB contenus dans les registres R1 et R2 et donne cette somme en DCB dans R3. On indiquera dans un registre annexe si cette somme est supérieure ou non à 10^8

On indiquera auparavant l'algorithme utilisé pour faire ce calcul.

Question 2: (4 points) Synthèse d'automate

On veut détecter tous les passages de 0 à 1 et de 1 à 0 dans une séquence quelconque de 1 et de 0.
Le système à concevoir possède donc une entrée **e** et une sortie **s** qui prend la valeur 1 après tout passage de **e** de 0 à 1 ou de 1 à 0.

Exemple :

e : 0 0 1 1 0 1 1 1 0 0 1 0 0 0
s : 0 0 1 0 1 1 0 0 1 0 1 1 0 0

1. Donner le graphe de l'automate d'états finis qui modélise ce système. On précisera le type de l'automate : Moore ou Mealy.

2. Donner le dessin du circuit réalisant cet automate à l'aide de bascules D sensibles au front montant, de portes NAND et Inverseurs.

3. Compléter le chronogramme de l'annexe A en donnant les valeurs de **s, Etat** (la valeur de l'état courant en sortie des bascules) et **NouvelEtat** (la valeur du nouvel état en entrée des bascules). **e** est l'entrée de ce circuit, **s** la sortie, **H** l'horloge activant la (ou les) bascules D, **Init** le signal d'initialisation du système.
   On pourra rendre l'annexe avec la copie d'examen.

Question 3: (6 points)

Voici le début et la fin d'un programme en langage C:

```
#include <stdio.h>

void main ()
{
unsigned int a=19,b=7, x, q, r;

x=1;
while (…)
…

printf("Resultats: %d %d", q, r);
}
```

Voici le résultat en assembleur ARM de la compilation de ce programme (gcc -S prog.c):

```
        .file   "prog.c"
        .section        .rodata
        .align  2
.LC0:
        .ascii  "Resultats: %d %d\000"
        .text
        .align  2
        .global         main
        .type   main,function
main:
        @ args = 0, pretend = 0, frame = 20
        @ frame_needed = 1, uses_anonymous_args = 0
        mov     ip, sp
        stmfd   sp!, {fp, ip, lr, pc}
        sub     fp, ip, #4
        sub     sp, sp, #20
        mov     r3, #19
        str     r3, [fp, #-16]
        mov     r3, #7
        str     r3, [fp, #-20]
        mov     r3, #1
        str     r3, [fp, #-24]
.L2:
        ldr     r2, [fp, #-24]
        ldr     r3, [fp, #-20]
        mul     r2, r3, r2
        ldr     r3, [fp, #-16]
        cmp     r2, r3
        bls     .L4
        b       .L3
.L4:
        ldr     r3, [fp, #-24]
        add     r3, r3, #1
        str     r3, [fp, #-24]
        b       .L2
.L3:
        ldr     r3, [fp, #-24]
        sub     r3, r3, #1
        str     r3, [fp, #-28]
        ldr     r2, [fp, #-28]
        ldr     r3, [fp, #-20]
        mul     r2, r3, r2
        ldr     r3, [fp, #-16]
        rsb     r3, r2, r3
        str     r3, [fp, #-32]
.L7⊡
        ldr     r0, .L5
        ldr     r1, [fp, #-28]
        ldr     r2, [fp, #-32]
        bl      printf
        ldmea fp, {fp, sp, pc}
.L6:
        .align  2
.L5:
        .word  .LC0
.Lfe1:
        .size   main,.Lfe1-main
        .ident  "GCC: (GNU) 3.1"
```

RAPPELS

Les instructions assembleur du programme commencent à l'étiquette **main.**

sp est le registre pointeur de sommet de pile

fp est le pointeur de la base du contexte d'une procédure

rsb r1, r2, r3 effectue une soustraction inverse (r1=r3-r2)

mul effectue une multiplication.

ldr r0,.L5 est équivalent à **ldr r0, =.L5**

unsigned int est le type entier non signé sur 32 bits.

printf est une fonction de la bibliothèque d'entrée/sortie standard du langage C permettant d'afficher une chaîne de caractère à l'écran. Le premier paramètre est une chaîne de caractères. Les paramètres suivants dépendent du contenu du premier paramètre. Une occurrence de %d dans la chaîne implique un paramètre de type entier supplémentaire. C'est la valeur de ce paramètre entier qui est affichée à l'écran à la place de %d.

Répondre aux questions suivantes

1. A quoi servent les instructions **mov ip, sp**,

stmfd sp!, {fp, ip, lr, pc} et **ldmea fp, {fp, sp, pc}** ?

Quels sont leurs effets ?

2. Où sont stockées les variables du programme ?

   Représenter l'état de la pile lors du passage à l'étiquette **L3**:

   - Pointeurs sp et fp
   - Emplacement des variables du programme

3. Quelle est la condition du while apparaissant dans le programme ? Quelles sont les étiquettes utilisées pour traduire le while ? Expliquez comment est traduit le while.

4. Comment sait-on à la vue du programme en ARM que les variables sont des entiers non signés ?

5. Compléter le programme C. Quel est le contenu des variables q et r à la fin du programme.

6. Les instructions qui suivent l'étiquette **.L7** sont la traduction de l'appel à la fonction printf : « printf("Resultats: %d,%d \n",q, r) ».

   Où sont stockés les 3 paramètres de la procédure printf ?

   Quel est l'effet de l'instruction **bl printf** ?

7. Quelles sont les valeurs de ces trois paramètres lors de l'exécution du programme au moment de l'appel à printf ? A quoi sert l'étiquette **LC0** ?

Question 4: (6 points) Machine algorithmique

On veut réaliser un circuit permettant d'effectuer la division entière d'un entier positif A par un entier positif B. Le résultat de ce calcul est le quotient Q et le reste R de cette division : A=B.Q +R avec 0≤R<A.

Soit l'algorithme suivant qui effectue ce calcul :

```
Lexique :
        A et B : deux entiers > 0 ; { les données } ;
        Q,R : deux entiers ≥ 0; { les résultats }
Algorithme :
        X=B ; Q=1 ;
        Tant que X≤A faire
        Debut
                X=X+B ;
                Q=Q+1 ;
        Fin
        Q=Q-1 ;
        R=A-(X-B) ;
```

On utilise une architecture « partie opérative/partie contrôle » pour ce circuit. On ne s'intéresse pas aux entrées/sorties du circuit. A et B sont donnés dans deux registres. On veut avoir Q et R dans deux registres spécifiques en fin de calcul.

1- On veut utiliser une UAL à deux opérandes. Transformez l'algorithme afin que n'apparaissent que des opérations à deux opérandes.

2- Enumérez les opérations et les variables apparaissant dans cet algorithme. Quels sont les comptes-rendus que la PO devra envoyer à la partie contrôle ?

En déduire une partie opérative permettant de réalisez tous les calculs et affectations de variables apparaissant dans cet algorithme. On donnera un nom à chacun des signaux de commandes de cette PO. On précisera le codage des commandes de l'UAL pour chacune des opérations qu'il effectue.

3- Donnez le graphe de l'automate commandant la PO précédemment donnée.
On précisera les entrées et les sorties de cet automate. On détaillera la valeur de ces entrées et sorties pour chaque état. On ne réalisera pas le circuit correspondant.

ANNEXE⊡

Chronogrammes