DST UE21 - M2101
Architecture et programmation des mécanismes de base d'un système informatique

IUT de Paris Descartes, Département Informatique
Ph. Darche

Mars 2015
Durée : 1h30

Aucun document autorisé. Calculatrice non autorisée. Sujet de deux pages et une annexe.
Les parties sont indépendantes. Répondez dans l'ordre de numérotation des problèmes. Barème indicatif.
Justifiez clairement vos réponses.

Partie I : Questions de cours (10 points)

Q1. A quoi servent les déclarations STACK, DATASEG et CODESEG d'un programme en langage d'assemblage ?

Q2. Les registres.

Q2.1. A quoi sert le compteur ordinal ? Comment s'appelle-t-il dans le microprocesseur 8086 (acronyme et nom complet) ?

Q2.2. Donnez la signification de l'acronyme RI et que contient ce registre ?

Q2.3. Qu'est-ce qu'un registre implicite ? Citez deux instructions qui en utilisent et le nom du ou des registres correspondants.

Q3. Donnez les trois étapes du cycle d'exécution de base d'une instruction et précisez, pour chacune d'elles, ce qu'elles réalisent.

Q4. Le sous-programme.

Q4.1. Quelle est la fonction de l'instruction *call* ? Expliquez en détail son exécution.

Q4.2. Quelle est la fonction de l'instruction *ret* ? Expliquez en détail son exécution.

Q4.3. A quoi peut servir la pile lorsque des sous-programmes sont utilisés ?

Q4.4. Quelles sont les deux formes de passage d'un paramètre ? Précisez les trois voies de passage possibles ?

Partie II : Le langage d'assemblage (10 points)

On décide d'écrire un programme en langage d'assemblage 8086 qui calcule le polynôme P défini par :

$P(x) = x^2 + 2x + 1$ avec $x \in \mathbb{N}$.

Nous allons décomposer le problème en deux parties. La première permettra de calculer le carré et la seconde, le restant des termes. **Vous éviterez au maximum d'utiliser les variables mémoires dans le programme pour une question de rapidité d'exécution.** Par contre, en fin de chaque programme, vous n'oublierez pas de ranger le résultat du calcul dans la variable concernée.

Partie 1. Calcul du carré.

Ecriture d'un programme en langage d'assemblage 8086 qui calcule le carré de x ($x \in \mathbb{N}$, $x < 256$ et rappelons que $255^2 = 65025$). **Un squelette de programme est fourni en annexe 1. Les parties 1 et 2 seront écrites sur votre copie.**

Q1. Deux variables, `x` et `résultat`, seront définies. Pour vous aider à définir les formats de ces variables, veuillez remplir le tableau QI-1.

Variables	Valeur minimale	Valeur maximale	Format minimum pour l'intervalle de validité (bits)	Format retenu pour le programme
`x`				
`résultat`				

Tableau QI-1 : Etude du format des variables pour le calcul de la fonction quadratique (*i.e.* x^2)

Q2. Donnez la définition des variables correspondantes en langage d'assemblage avec une initialisation. Précisez le numéro de la partie concernée dans le programme squelette. **Pour la suite de cette partie 1, aucune autre variable ne devra être déclarée.**

Q3. Ecrivez le programme correspondant appelé **avec interdiction d'utiliser un sous-programme**. Vous devrez par ailleurs utiliser l'instruction `mul`.

Q4. Transformation de votre programme en sous-programme avec passage obligatoire des paramètres par valeur et par registre.

Q4.1. Quel est l'avantage de passer les paramètres par les registres ?

Q4.2. Y a-t-il un inconvénient ou une limitation de les utiliser (*i.e.* les registres) ?

Q4.3. Combien de registres allez-vous utiliser pour ce passage ? Nommez-les ?

Q4.4. Ecrivez le sous-programme que vous appellerez *carré_sp* ainsi que le programme l'appelant (nouvelle partie 2). **Un squelette de sous-programme (partie 3) est fourni en annexe 2.**

Partie 2. Calcul du polynôme.

Q1. Le calcul de P(255) donne 65536 ($=2^{16}$). Sachant que le microprocesseur 8086 possède une architecture 16 bits, que va-t-il se passer et quelle sera la valeur rangée dans le registre qui recevra le résultat ?

Q2. Ecrivez le programme polynôme (nouvelles parties 1 et 2) qui calcule P(x) avec x, entier naturel < 256 en utilisant le sous-programme de la question 4 de la partie 1 qui calcule x^2. **Il est donc inutile de ré-écrire ce dernier.** N'oubliez pas de traiter le cas particulier x = 255 (*cf.* la question précédente).

Annexe 1 : Squelette de programme en langage d'assemblage 8086

```
; Nom du programme : XX.asm
; Fonction : XX à remplir

IDEAL
DOSSEG
MODEL TINY
P8086

STACK 100h

DATASEG

        Partie 1


CODESEG

; programme principal

debut:  mov ax,@data
        mov ds,ax
        mov es,ax

; corps du programme

        Partie 2

        mov ah,4ch
        int 21h
END debut
```

Annexe 2 : Squelette de sous-programme

```
; déclaration du sous-programme

PROC carré_sp NEAR

        Partie 3

fin_sp:

ENDP carré_sp
```

Saut sur état d'indicateur

- Retenue (CF)
  - JNC (*Jump on Not Carry*)
  - JAE (*Jump on Above or Equal*)
  - JNB (*Jump on Not Below*)
    - CF = 0
  - JC (*Jump on Carry*)
  - JB (*Jump on Below*)
  - JNAE (*Jump on Not Above or Equal*)
    - CF = 1

Saut sur état d'indicateur

- Zéro (ZF)
  - JNE (*Jump on Not Equal*)
  - JNZ (*Jump on Not Zero*)
    - ZF= 0
  - JE (*Jump on Equal*)
  - JZ (*Jump on Zero*)
    - ZF= 1

Saut sur état d'indicateur

- Dépassement de capacité (OF)
  - JNO (*Jump on Not Overflow*)
    - OF = 0
  - JO (*Jump on Overflow*)
    - OF = 1
- Parité (PF)
  - JNP (*Jump on Not Parity*)
  - JPO (*Jump on Parity Odd*)
    - PF = 0
  - JP (*Jump on Parity*)
  - JPE (*Jump on Parity Even*)
    - PF = 1

Saut sur état d'indicateur

- Signe (SF)
  - JNS (*Jump on Not Sign*)
    - SF = 0
  - JS (*Jump on Sign*)
    - SF = 1

Saut sur état d'indicateur

- Représentation non signée
  - supérieur (>)
    - JA (*Jump on Above*)
    - JNBE (*Jump on Not Below or Equal*)
      - (CF and ZF) = 0
  - supérieur ou égal (≥)
    - JAE (*Jump on Above or Equal*)
    - JNB (*Jump on Not Below*)
      - CF = 0
  - inférieur (<)
    - JB (*Jump on Below*)
    - JNAE (*Jump on Not Above or Equal*)
      - CF = 1

Saut sur état d'indicateur

- Représentation non signée (suite)
  - inférieur ou égal (≤)
    - JBE (*Jump on Below or Equal*)
    - JNA (*Jump on Not Above*)
      - (CF or ZF) = 1
  - non égal (≠)
    - JNE (*Jump on Not Equal*)
    - JNZ (*Jump on Not Zero*)
    - ZF = 0
  - égal (=)
    - JE (*Jump on Equal*)
    - JZ (*Jump on Zero*)
      - ZF = 1