# Travaux Pratiques d'assembleur

Les TP d'Assembleur vous permettront de programmer le simulateur du microprocesseur présenté en cours et en TD. Vous devez faire valider l'avancée de votre travail par l'encadrant au moins une fois par séance et systématiquement à la fin de chaque exercice.

**Tous les codes devront être indentés et suffisamment commentés pour être compréhensibles.**

## Initiation à la programmation du simulateur

### Exercice 1: Fiche de cours à rendre en rentrant dans la salle

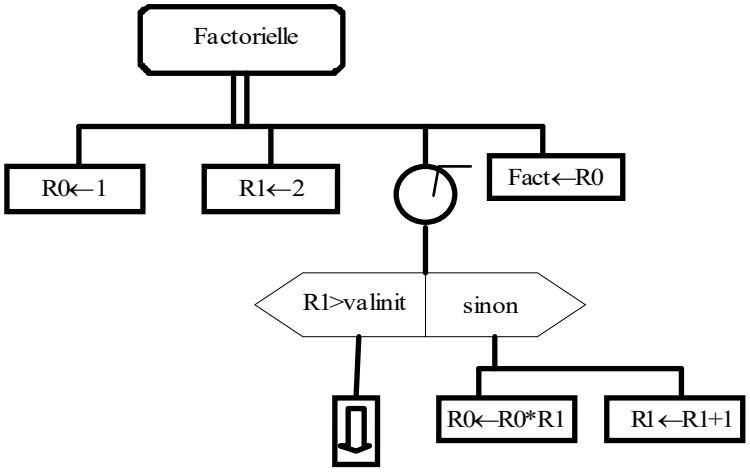
1. Parcourez le guide de programmation du simulateur de processeur en particulier le dernier paragraphe intitulé « Utiliser le simulateur ».
2. A partir des fichiers du cours et du guide de programmation, concevez un document texte d'une page résumant les informations essentielles nécessaires à vos débuts dans la programmation avec le simulateur.

### Exercice 2: Tests des fichiers de démonstrations

1. A partir du cours d'elearn, copiez sur votre espace de travail le cours, le guide de programmation, ce sujet de TP, le fichier de l'exécutable du simulateur de microcontroleur et le répertoire des fichiers de démonstration. Lancez le simulateur en double cliquant sur l'exécutable ou à l'aide de la fenêtre de commande DOS.
2. Chargez le programme balle.mpc puis exécutez-le.
3. Rechargez le programme. A partir de l'affichage de la zone du microprocesseur dans le simulateur, identifiez à quelle adresse mémoire se trouve la première instruction du code.
4. Visualisez cette première instruction dans la mémoire. En vous remémorant les codes du cours et des TD, définissez le code écrit par le programmeur pour obtenir cette instruction en mémoire. Déterminez l'adresse de la pile et visualisez-la.
5. Lancez le programme en mode pas à pas avec une vitesse d'exécution faible et suivez l'évolution des registres et de la mémoire en fonction des instructions en cours de réalisation à l'aide du code couleur rouge/vert.
6. Que contient SP à la fin de la première instruction ? Pourquoi ?
7. Vérifiez le fonctionnement des 4 premières instructions du programme en les exécutant pas à pas à vitesse faible puis exécutez normalement le programme.
8. Expliquez pourquoi Mem(101)=14 à la fin de ces 4 instructions. (voir diapositive 98 du cours pour comprendre le fonctionnement du PUSH)
9. Testez les deux autres programmes de démonstration en les exécutant en mode pas à pas sur quelques instructions et en suivant l'évolution des registres et de la mémoire en fonction des instructions en cours de réalisation. Testez-les en vitesse normale.

## Exercice 3: Factoriel

L'objectif de cet exercice est d'écrire un programme en assembleur qui calcule la factorielle de 6. Il respecte l'algorithme suivant :




Ce programme déclare une variable 'valinit' initialisée à 6 et une variable 'Fact' non initialisée.

Vous pourrez utiliser un traitement de texte quelconque ou un EDI (wordpad, bloc note, Crimson Editor…).

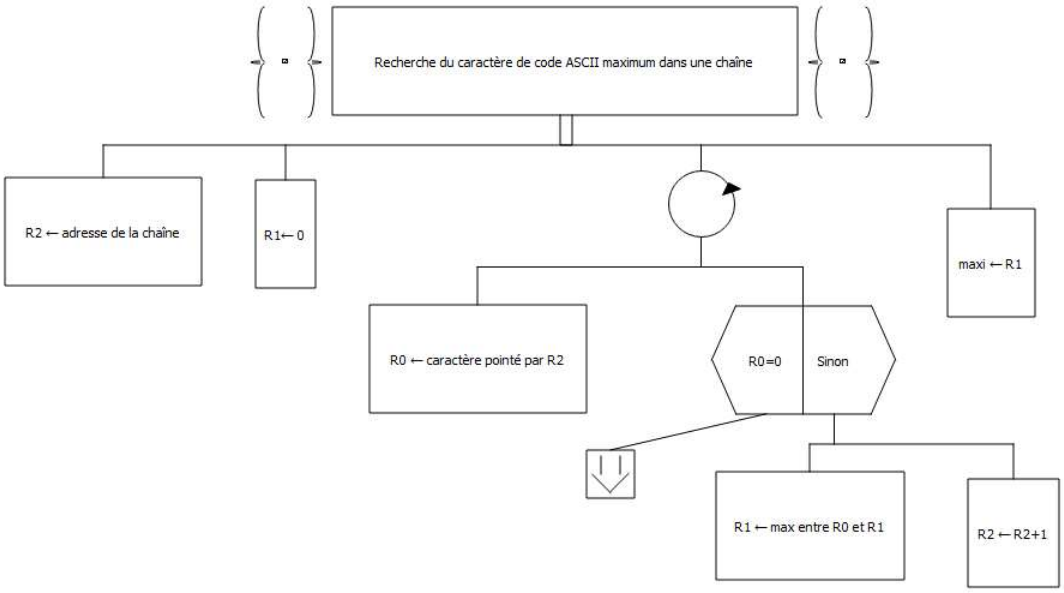
1. Etudiez l'algorithme afin d'en comprendre le fonctionnement.
2. Identifiez le test qui vous permettra de valider le programme réalisant cet algorithme.
3. Ecrivez le programme en assembleur. Si vous utilisez le Bloc Note veillez à ce que l'extension soit correcte si besoin en décochant « masquer l'extension des fichiers connus » dans Options des dossiers de l'explorateur
4. Lancez le simulateur et compiler ce programme. Lorsque la compilation est réussie, charger le programme en mémoire. Ne pas oublier de choisir pour la variable 'Fact' un affichage en entier naturel (en cliquant sur sa valeur en mémoire pour accéder au choix du mode d'affichage), puis exécuter le programme en pas à pas de façon à voir comment évolue le contenu de la variable 'Fact'. Vérifiez le résultat obtenu avec le test de la question 2.
5. Copiez le programme dans un nouveau fichier que vous renommerez puis modifiez ce nouveau programme pour que la variable 'valinit' soit initialisée à 13. Exécutez le programme. Le résultat obtenu est-il correct ? Pourquoi ? Identifiez à quel moment et de quelle manière le simulateur vous indique ces informations en vous aidant de la description du registre d'état donnée en page 4 du guide de programmation et du cours sur ce même sujet.

## Exercice 4: Chaîne de caractères

Cet exercice a pour but de vous faire écrire un programme qui:

- déclare une chaîne de caractères avec un texte de votre choix
- recherche dans cette chaîne le caractère ayant le code ASCII le plus grand
- place celui-ci dans une variable appelée 'maxi'.

La chaîne de caractères sera terminée par 0 (pas le caractère '0' mais le chiffre 0). Le programme devra suivre l'algorithme ci-dessous :




1. Etudiez l'algorithme afin d'en comprendre le fonctionnement.
2. Identifiez le test qui vous permettra de valider le programme réalisant cet algorithme.
3. Ecrivez le programme.
4. Chargez ce programme en mémoire et choisissez pour la variable 'maxi' un mode d'affichage en caractère. Exécutez le programme et vérifiez que, lorsqu'il se termine, on a bien dans 'maxi' le caractère de la chaîne choisie le plus loin dans l'alphabet : pour cela, choisissez un affichage sous forme de nombre pour les caractères de la chaine et le résultat en cliquant sur Mode d'affichage dans la zone de visualisation de la Mémoire.

## Affichage de texte

### Exercice 5: Soulignement de texte

1. Ecrire un programme qui efface l'écran puis y affiche un texte. Le texte à afficher sera placé dans une variable initialisée de type chaîne de caractères terminée par un point. Le programme parcourt la chaîne caractère par caractère et affiche chaque caractère à l'écran jusqu'à rencontrer le point '.' qui ne sera pas affiché.

Notez que lorsque l'on écrit un caractère sur le périphérique du simulateur, les contenus des ports 1 et 2 sont automatiquement mis à jour pour pouvoir écrire le caractère suivant à l'endroit adéquat (voir guide de programmation page 6).

2. Identifiez le test qui vous permettra de valider le programme.
3. Testez le programme.
4. Complétez le programme pour que le texte soit ensuite souligné.

Vous noterez que quand on écrit un caractère à l'écran la coordonnée en x contenue dans Port 1 est mise à jour donc on peut savoir jusqu'où doit aller la ligne qui sert à souligner en relevant la valeur du Port 1 à la fin de

l'écriture du texte. Le soulignement devra être visible à l'écran et donc suffisamment éloigné du texte pour ne pas le toucher sans en être trop éloigné.

5. Identifiez le test qui vous permettra de valider le programme.
6. Testez le programme.

## Utilisation des touches et de la souris

### Exercice 6: Dessin conditionné par l'utilisateur

1. Ecrire un programme qui :

- dessine un carré de 100 pixels de côté au centre de l'écran
- chaque fois que l'on appuie sur la touche A, le carré se réduit à 50 pixels de côté
- chaque fois que l'on lâche la touche A, le carré retrouve sa taille initiale
- lorsque l'on appuie sur la touche B, le programme se termine

Pour écrire ce programme vous définirez 2 procédures : l'une qui dessine un carré de 100 pixels de côté au centre de l'écran, l'autre qui efface l'écran puis y dessine un carré de 50 pixels de côté au centre.

2. Identifiez le test qui vous permettra de valider le programme.
3. Testez le programme.

### Exercice 7: Attente d'une action de l'utilisateur

1. Modifier le programme de l'exercice précédent pour, qu'au début, il attende un clic de souris pour placer le carré de 100 pixels de côté centré sur le point où a eu lieu ce clic. Après quoi il fonctionne comme décrit à l'exercice précédent : réduction du carré, rétablissement et sortie du programme.
2. Identifiez le test qui vous permettra de valider le programme.
3. Testez le programme.

## Dessins

### Exercice 8: Dessin d'un rectangle

1. Ecrire une procédure de dessin d'un rectangle plein qui reçoit dans la pile les 5 paramètres suivants et qui affiche le rectangle à l'écran.
    - un code de couleur (entier de 0 à 15)
    - la coordonnée en x du coin supérieur gauche du rectangle
    - la coordonnée en y du coin supérieur gauche du rectangle
    - la largeur du rectangle
    - la hauteur du rectangle
2. Ecrire un programme permettant de tester la procédure.
3. Identifiez le test qui vous permettra de valider le programme.
4. Testez le programme.

### Exercice 9: Mire

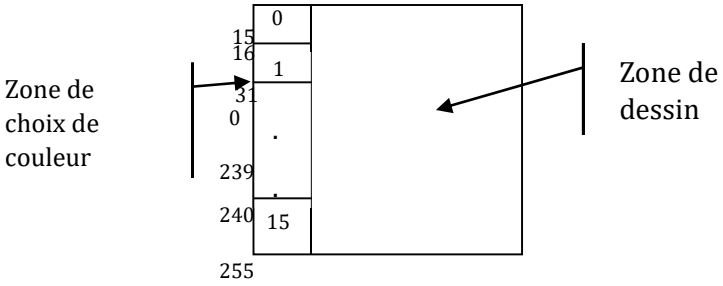
1. Ecrire un programme de test de l'écran qui utilise cette procédure pour dessiner une mire. Cette mire sera constituée de 16 bandes horizontales de couleurs différentes occupant tout l'écran. Chaque bande aura donc une largeur de 255 pour une hauteur de 16.

2. Identifiez le test qui vous permettra de valider le programme.
3. Testez le programme.

## Dessin à la souris

### Exercice 10: Mini Draw

L'objectif est d'écrire un petit programme de dessin à la souris. L'écran se présente comme suit :

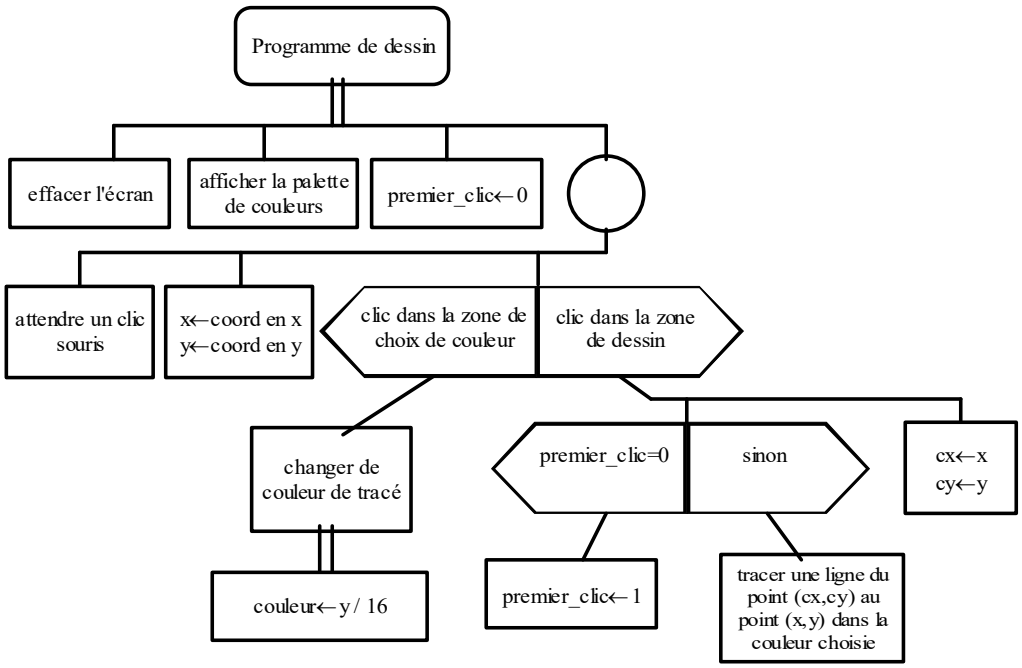



Dans la zone de choix de couleur on voit 16 rectangles de 20x16 chacun d'une couleur différente.

On cliquera sur l'un de ces rectangles pour choisir une couleur de tracé. Comme les rectangles représentant les couleurs ont une hauteur de 16 pixels, le numéro de la couleur correspondant à un rectangle peut être obtenu en divisant par 16 (division entière) la coordonnée en y de la souris lorsque l'on clique sur un choix (les couleurs sont codées entre 0 et 15).

Quand on clique dans la zone de dessin, une ligne est tracée entre le point où l'on vient de cliquer et le point précédent où l'on avait cliqué. Cette ligne est tracée dans la dernière couleur choisie. Le premier clic dans cette zone n'a donc aucun effet visible, la première ligne sera tracée entre ce point et le deuxième clic.

L'algorithme de ce programme est le suivant :



Remarque : pour savoir si un clic a eu lieu dans la zone de choix de couleur ou dans la zone de dessin, il suffit de regarder la coordonnée en x du clic : si elle est inférieure ou égale à 20 (largeur des rectangles représentant les couleurs) on est en zone de choix de couleur, sinon on est en zone de dessin.

1. Etudiez l'algorithme afin d'en comprendre le fonctionnement.
2. Réalisez le programme demandé.
3. Identifiez le test qui vous permettra de valider le programme.
4. Testez le programme.
5. Ajoutez au programme précédent la gestion des touches A et B de façon à ce que le bouton A permette d'effacer la zone de dessin et de réinitialiser premier_clic et le bouton B termine le programme.
6. Identifiez le test qui vous permettra de valider le programme.
7. Testez le programme.