

## Objet du TP

Vous allez développer un petit programme (calcul de PGCD) en assembleur Nios. Pour compiler et charger vos programmes sur carte FPGA, vous utiliserez **Altera Monitor**.

## Prise en main d'Altera Monitor

Dans cette section, vous allez compiler un premier programme, le charger sur carte FPGA puis contrôler son exécution depuis Altera Monitor. Vous vous initiez ainsi aux fonctions de base du logiciel, que vous utiliserez dans la seconde partie et tout au long des deux prochains TP.

Précision importante : contrairement à (par exemple) Eclipse, **Altera Monitor** n'est pas un environnement de développement intégré et ne contient pas d'éditeur de code. Pour écrire vos programmes, vous utiliserez un logiciel auxiliaire. Nous vous suggérons **Notepad++**, installé sur vos machines, mais le bloc-notes de Windows convient également.

## Création du projet

Chaque logiciel développé sous Altera Monitor forme un **projet**. Le moniteur fonctionne sur un seul projet à la fois et conserve toutes les informations sur ce projet dans un répertoire unique.

Pour ce TP, vous partirez d'un squelette de projet que vous trouverez au chemin suivant :

**share:\I3info\ILM\TP\TP1\PGCD**

**IMPORTANT** : Copiez le répertoire PGCD dans votre espace personnel. Par exemple à l'emplacement **H:\ILM\TP\TP1** (la suite suppose que vous avez fait ce choix).

Le disque *share* : étant en lecture seule, vous rencontrerez des erreurs si vous omettez cette étape.

Le seul fichier qui présente un intérêt pour nous est le fichier **main.s** : il comporte le code source du programme. Vous pouvez dès à présent l'ouvrir dans votre éditeur de code.

Prenez le temps de vous familiariser avec le code du programme avant de poursuivre.

## Lancement d'Altera Monitor

Pour lancer Altera Monitor, nous vous demandons d'exécuter le fichier **.bat** suivant :

**share:\I3info\ILM\TP\monitor.bat**

**IMPORTANT**: Ne lancez **pas** Altera Monitor depuis le menu Démarrer. Passer par ce script est nécessaire pour éviter des erreurs lors de la compilation de vos programmes.

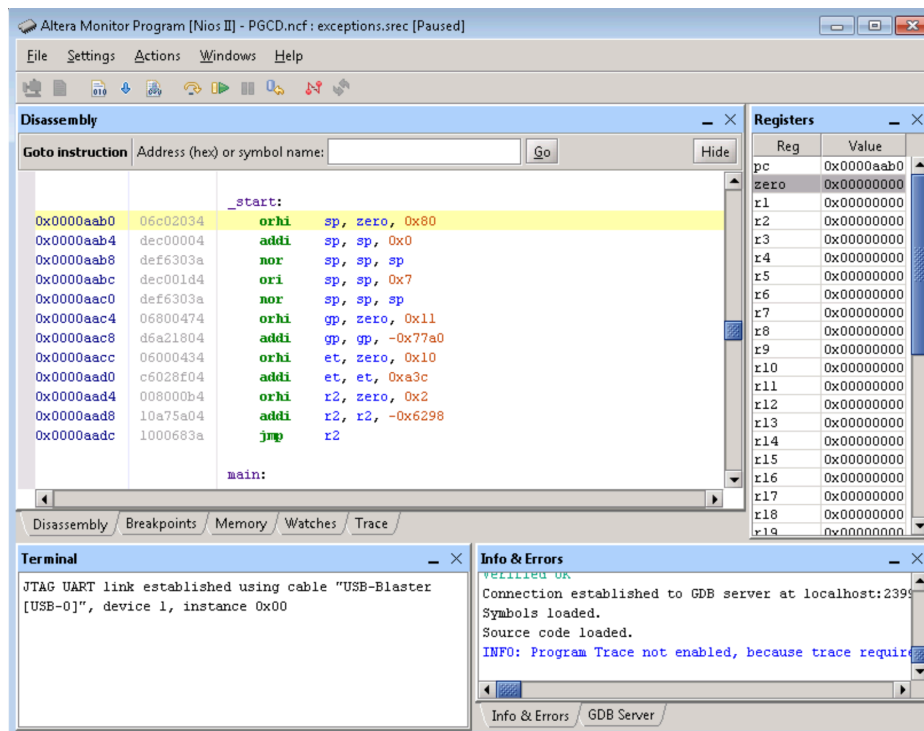
Une fois le moniteur lancé, ouvrez votre projet:

**H:\ILM\TP\TP1\PGCD\PGCD.amp**

Une boîte de dialogue apparaît. Vérifiez que la carte FPGA est branchée sur secteur et allumée, puis cliquez sur *Oui* afin de charger le processeur *Nios II* sur celle-ci.

## Compilation

Une fois l'opération terminée, appuyez sur la touche **F5** pour compiler votre projet et charger le code exécutable sur la carte. Si tout se passe bien, votre écran devrait maintenant ressembler à l'image suivante :



## Présentation de l'interface

L'interface est divisée en quatre zones :

- La **zone principale** comporte plusieurs onglets fournissant diverses fonctionnalités pour déboguer vos programmes. Nous nous intéresserons à deux d'entre eux :

- **Disassembly** : C'est l'onglet que vous utiliserez le plus souvent. Cette vue présente les instructions machine exécutées par le processeur.

Pour la commodité, le début des fonctions et sous-procédures est indiqué par le symbole correspondant. Il est possible de rechercher un symbole particulier en entrant son nom dans le champ : *Address (hex) or symbol name* et en cliquant sur *Go*.

- **Memory** : Dans cet onglet, vous pouvez voir le contenu brut de la mémoire. La section *.data* débute à l'adresse 0x17384. Vous pourrez y observer les modifications effectuées par votre programme. Nous y reviendrons.

- **Registers** : Cette zone liste les registres du processeur et leurs valeurs à un instant *t*. Elle vous sera utile pour déboguer vos programmes en exécution pas-à-pas et quand vous utiliserez des points d'arrêt.

- **Terminal** : C'est l'interface d'E/S par défaut de la carte. C'est là que les messages générés par la carte seront affichés, et c'est là aussi que vous saisissez les valeurs demandées par votre programme.

- **Info & Errors** : En cas de problème de compilation, consultez toujours cette zone. Les messages d'erreur sont affichés en rouge.

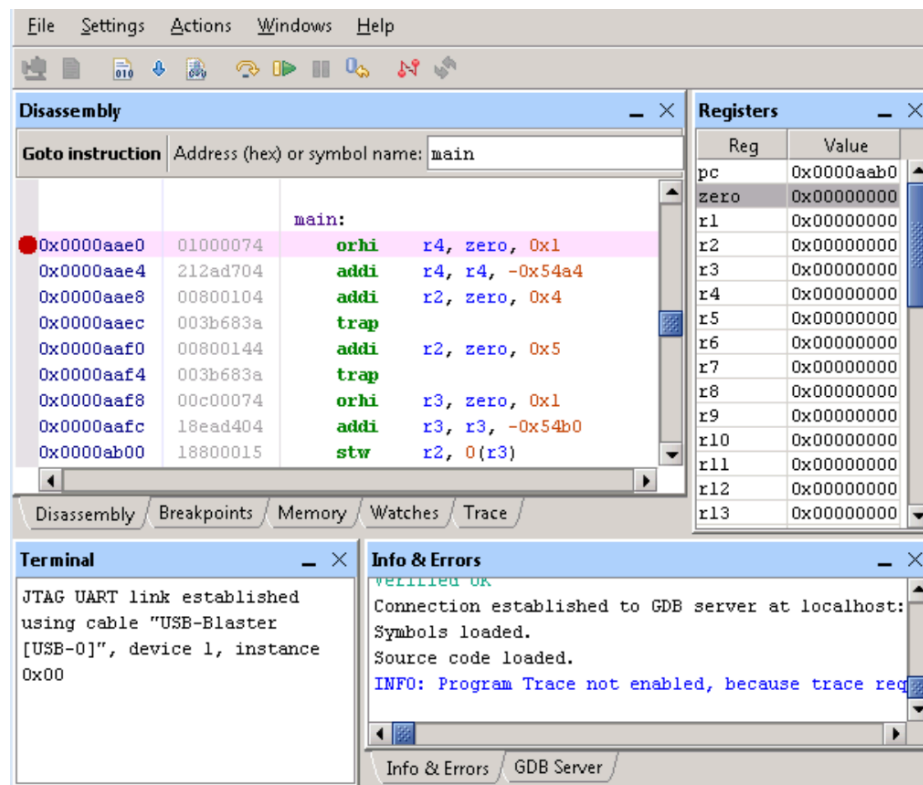
## Exécution et débogage


Le programme étant compilé et chargé, nous allons maintenant l'exécuter.

Avant de passer la main à la fonction **main** qui constitue le point d'entrée de votre programme, la carte passe par une phase d'initialisation qui n'a pas d'intérêt pour nous. C'est pourquoi nous vous demandons de mettre en place un point d'arrêt (*breakpoint*) au début de la fonction **main**. Pour cela :



- Cherchez le symbole **main** en saisissant son nom dans le champ "Address or symbol name".
- Cliquez sur la barre grise à gauche de la première instruction de la fonction.


Un point rouge dans cette zone indique la présence du point d'arrêt.



Pour démarrer l'exécution de votre programme, cliquez sur le bouton  dans la barre d'outils. L'exécution devrait s'interrompre automatiquement au début de la fonction **main**.

Pour tester et déboguer votre programme, vous pouvez :

- Poursuivre son exécution en appuyant sur le bouton . Vous pouvez mettre en place de nouveaux points d'arrêt pour interrompre à nouveau l'exécution du programme à l'endroit de votre choix.
- Effectuer une exécution pas-à-pas en appuyant sur . Cette action exécute seulement l'instruction surlignée.

**Avertissement concernant l'instruction `trap` :** L'exécution pas-à-pas de l'instruction `trap` vous emmènera dans l'implémentation de l'appel système et n'apportera que de la confusion. Nous vous demandons plutôt de mettre en place un point d'arrêt **après** l'instruction et de poursuivre l'exécution du programme avec .

- Réinitialiser l'exécution en cliquant sur .

En cliquant sur , vérifiez que le comportement du programme est conforme à vos attentes.

## Recompilation

Après avoir modifié votre programme (ou si vous êtes perdus...) vous pouvez appuyer sur la touche **F5** pour recompiler votre programme et charger le nouveau code exécutable sur la carte.

Si une session de débogage est en cours, un dialogue apparaît. Cliquez simplement sur *Oui* pour interrompre l'exécution du programme et poursuivre l'opération. La compilation et le chargement du programme prend quelques secondes.

**Attention:** La recompilation du programme efface les points d'arrêt précédemment mis en place. Ceci est bien sûr valable pour le point d'arrêt sur la fonction *main* qui devra être ajouté à chaque recompilation.

## Exercice 1 : Inspection des registres et de la mémoire

1. La section de données *.data* débutant à l'adresse 0x17384, calculez l'adresse de la chaîne *invite\_a*.
2. Vérifiez cette valeur en inspectant la valeur du registre *r4* au moment opportun lors de l'exécution du programme (utilisez un point d'arrêt).
3. Cliquez sur l'onglet **Memory** dans la zone principale.
4. Saisissez l'adresse que vous avez déterminée en hexadécimal (sans le préfixe 0x) dans le champ "Address" puis cliquez sur "Go".
5. Faites un clic droit dans la zone, puis sélectionnez "Show equivalent ASCII characters." Reconnaissez-vous directement la chaîne *invite\_a* ? Pourquoi ?
6. Faites un clic droit dans la zone, puis sélectionnez *View as -> Byte*. Observez l'effet de ce changement, puis revenez en mode *Word*.

## Exercice 2 : Algorithme de calcul de PGCD

L'algorithme ci-dessous réalise le calcul du PGCD de deux nombres entiers positifs par la méthode des soustractions successives. Il est notoirement inefficace, mais peut être implémenté sur des architectures ne possédant pas d'opérateur de division.

```
void main() {  
    int x,y,res ;  
    print("Entrez un entier (a): ");  
    x= read_int() ;  
    print("Entrez un entier (b): ");  
    y= read_int() ;  
    while(x!=y) {  
        if (y>x) {  
            y= y-x;  
        } else {  
            x= x-y;  
        }  
    }  
    res = x;  
    print("Le pgcd de a et b est: %d ",res);  
}
```

Complétez le squelette de programme proposé, en respectant les consignes données en CM. Veillez en particulier à mettre en correspondance les instructions du langage de description avec les instructions machine utilisées.

Pour les entrées / sorties, vous aurez recours aux appels systèmes présentés en cours. Les appels à la fonction *print* seront implémentés avec *print\_int* et *print\_string*.

Vous exécuterez le programme en mode pas-à-pas ou utiliserez les points d'arrêt de manière à vérifier le bon fonctionnement de l'algorithme, en observant l'évolution des registres et de la mémoire. Le résultat, stocké en **mémoire** dans un mot, sera observé en fin d'exécution.

### Exercice 3 : Sous-programme de calcul du PGCD

Vous allez modifier le programme précédent en utilisant un sous-programme :

**int PGCD(int a, int b)**

appelé dans le programme principal, et dont le mode d'appel est le suivant :

- *a* sera placé dans r4,
- *b* sera placé dans r5,
- le résultat sera stocké dans r2.

On veillera à utiliser la pile pour préserver la valeur des registres dans le programme appelant, selon les conventions vues en cours.

```
int PGCD(int a,int b) {  
    while(a!=b) {  
        if (a>b) {  
            a= a-b ;  
        } else {  
            b= b-a ;  
        }  
    }  
    return a ;  
}
```