Curso de Graduação em Engenharia Eletrônica – Faculdade Gama – UnB
Eletrônica Embarcada – Código: 120871 - (Período 1/2019)
Professor: Gilmar Silva Beserra
E-mail: gilmar.beserra@gmail.com (Sala 24 – UED)

Atividade Prática 1[1]

OBJETIVO:

Escrever, montar e executar/depurar os primeiros programas em Assembly usando o CCS (*Code Composer Studio*).

INTRODUÇÃO:

O CCS é um ambiente integrado que oferece facilidades para se trabalhar com as diferentes fases do desenvolvimento de um sistema embarcado, listadas a seguir e mostradas no diagrama da Fig. 1.

- Projeto
- Preparação do Código
- Depuração (Debug) e
- Análise

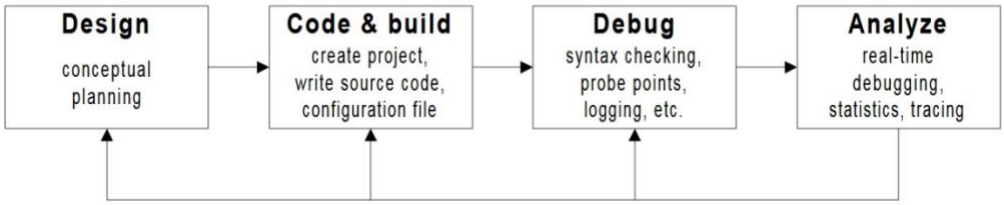

Fig. 1

Para criar um projeto em Assembly, basta seguir as etapas listadas abaixo.

1) Clique nas opções: "**File**", "**New**..." e "**Project**".
2) Na nova janela "**New Project**", clique na opção "**CCS Project**" e clique "**Next**".
3) Na nova janela "**New CCS Project**", verifique o nome do processador ("**MSP430G2553**"). Caso não esteja correto, digite o nome correto. Selecione a opção "**Empty Assembly only Project**", defina um nome para o programa e clique em finalizar (Fig. 2).

Após essas etapas, deverá surgir a tela de edição com o "esqueleto" do programa, mostrado abaixo. Seu código deverá ser digitado logo abaixo do campo "**Main loop here**". Nas próximas aulas, comentaremos o porquê das demais linhas de programa que aparecem neste arquivo.

[1] **Referência**: Módulo 0 do Laboratório de Sistemas Microprocessados (ENE-UnB)

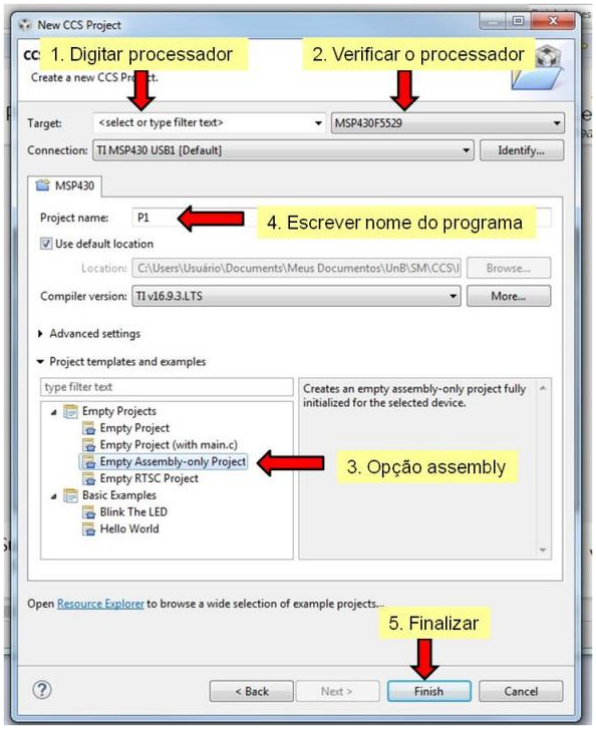

Fig. 2

```
;-------------------------------------------------------------------------------
; MSP430 Assembler Code Template for use with TI Code Composer Studio
;
;
;-------------------------------------------------------------------------------
        .cdecls C,LIST,"msp430.h"       ; Include device header file

;-------------------------------------------------------------------------------
        .def    RESET                   ; Export program entry-point to
                                        ; make it known to linker.
;-------------------------------------------------------------------------------
        .text                           ; Assemble into program memory.
        .retain                         ; Override ELF conditional linking
                                        ; and retain current section.
        .retainrefs                     ; And retain any sections that have
                                        ; references to current section.

;-------------------------------------------------------------------------------
RESET       mov.w   #__STACK_END,SP     ; Initialize stackpointer
StopWDT     mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer
```

```
;-------------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------------
                                        Seu código entra aqui.
;-------------------------------------------------------------------------
; Stack Pointer definition
;-------------------------------------------------------------------------
        .global __STACK_END
        .sect   .stack

;-------------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------------
        .sect   ".reset"            ; MSP430 RESET Vector
        .short  RESET
```

Atalhos interessantes do Code Composer:

CTRL + S	Salva o programa
F11	Monta o programa e ativa o ambiente de debug
F5	Executa passo a passo (Step Into);
F6	Executa passo a passo, mas não "entra" nas sub-rotinas (Step Over);
F8	Executa (roda) o programa
ALT + F8	Pausa execução
CTRL + SHIFT + R	Soft Reset
CTRL + F2	Termina o debug e volta para o editor
CTRL + SHIFT + B	Insere ou remove um Ponto de Quebra (Break Point)

TAREFAS:

A) Escreva, monte e execute com o debug o programa listado abaixo (**P1A**), que inicializa o conteúdo dos registradores **R5** e **R6** e depois os soma, guardando o resultado em **R6**. Note que ao lado de cada mnemônico foi colocado o modificador "**.B**", que indica que a instrução opera em 8 bits.

Usando **F5**, execute o programa passo a passo. Para ver o conteúdo dos registradores, selecione as opções "**View**" e "**Registers**" na barra de ferramentas. Na nova janela, clique no pequeno triângulo à esquerda da linha "**Core Registers**". Clicando com o botão da direita sobre o registrador, você pode usar a opção "**Number Format**" para mudar a apresentação do número. A opção hexadecimal é a padrão.

```
;-------------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------------
;
P1A:    MOV.B       #3,R5           ;Colocar o número 3 em R5
        MOV.B       #4,R6           ;Colocar o número 4 em R6
        ADD.B       R5,R6           ;Fazer a operação R6 = R5 + R6
        JMP         $               ;Travar execução num laço infinito
        NOP                         ;Nenhuma operação
```

Ao escrever seus programas, use a tabulação para separar as 4 colunas:

Coluna 1: mais à esquerda, reservada para os rótulos (*labels*). No programa acima, "P1A" é um label que faz referência ao endereço da instrução **MOV.B #3,R5**. Sempre coloque seus *labels* nessa primeira coluna. Nunca coloque uma instrução nesta posição, pois ela será interpretada como *label*.

Coluna 2: onde são colocados os mnemônicos das instruções (MOV, ADD, JMP, etc).

Coluna 3: onde são colocados os operandos das instruções (#3,R5; R5,R6, etc). Note que a atribuição é da esquerda para a direita.

Coluna 4: destinada aos comentários. Tudo que estiver após o ponto e vírgula (**;**) é ignorado pelo Assembler. Note que o comentário pode iniciar na primeira coluna.

B) Escreva, monte e execute com o debug o programa **P1B**. Execute o programa **passo-a-passo** (usando a tecla **F5** ou **F6**). Note que o resultado em **R6** foi **0x4320**. Isso ocorre porque **0xFFFF** é a representação de -1 em complemento de 2. Você pode substituir a instrução **MOV #0xFFFF,R5** pela instrução **MOV #-1,R5,** que o resultado será o mesmo.

```
;-------------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------------
;
P1B:        MOV         #0xFFFF,R5      ;Colocar o número 0xFFF em R5
            MOV         #0x4321,R6      ;Colocar o número 0x4321 em R6
            ADD         R5,R6           ;Fazer a operação R6 = R5 + R6
            JMP         $               ;Travar execução num laço infinito
            NOP                         ;Nenhuma operação
```

C) Escreva, monte e execute com o debug o programa **P1C**, listado abaixo. Note que o programa usa a subrotina "SUBROT", que soma **1** ao **R5**, duas vezes. O número de vezes em que a subrotina é chamada é ditado pelo valor em **R6**. Isto significa que **R6** é o contador do laço (**LOOP**).

```
;-------------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------------
P1C:        CLR     R5          ;Zerar R5
            MOV     #4,R6       ;Colocar o número 4 em R6

LOOP:       CALL    #SUBROT     ;Chamar subrotina "SUBROT"
            DEC     R6          ;Decrementar R6
            JNZ     LOOP        ;Se diferente de zero, ir para LOOP
            NOP                 ;Nenhuma operação
            JMP     $           ;Travar execução num laço infinito
            NOP                 ;Nenhuma operação

SUBROT:     ADD     #1,R5       ;Somar 1 em R5
            ADD     #1,R5       ;Somar 1 em R5
            RET                 ;Retornar
```

Execute o programa até o final, usando **F5**. Faça o **Soft Reset** (CTRL + SHIFT + R) e execute o programa até o final usando **F6**. Você notou alguma diferença?

Vamos agora introduzir o conceito de **ponto de quebra** (***break point***). Quando a execução atinge um ponto de quebra, ela é interrompida e o controle é devolvido para o usuário. Coloque o cursor sobre a instrução **NOP**, logo antes do **JMP $** e ative um ponto de quebra neste local (**CTRL + SHIFT + B**). Uma forma alternativa para ativar o ponto de quebra é dar dois cliques no número que aparece à esquerda da instrução. O ponto de quebra é útil para rotinas longas ou demoradas.

Faça o **Soft Reset** (CTRL + SHIFT + R) e execute o programa com **F8**. Note que o programa é interrompido quando atinge este ponto.

D) Escreva, monte e execute com o debug o programa **P1D**, listado abaixo. Note que o programa usa a instrução **RLA R5,** que desloca o conteúdo de **R5** uma vez para a esquerda. A quantidade de rotações é dada por **R6**. O que aconteceu com o conteúdo de **R5** a cada laço de repetição? Você notou alguma coisa interessante?

```
;-------------------------------------------------------------------------
; Main loop here
;-------------------------------------------------------------------------
P1D:        MOV     #1,R5       ;Colocar 1 em R5
            MOV     #4,R6       ;Colocar o número 4 em R6
LOOP:       RLA     R5          ;Deslocar 1 bit para a esquerda
            DEC     R6          ;Decrementar R6
            JNZ     LOOP        ;Se diferente de zero, ir para LOOP
            NOP                 ;Nenhuma operação
            JMP     $           ;Travar execução num laço infinito
            NOP                 ;Nenhuma operação
```

E) Escreva, monte e execute com o debug o programa **P1E**, que deve acender os leds P1.0 e P1.6 da placa.

F) Escreva, monte e execute com o debug o programa **P1F**, que deve acender os leds P1.0 e P1.6 da placa apenas quando o botão P1.3 for pressionado.

G) Escreva, monte e execute com o debug o programa **P1G**, que deve calcular o oitavo termo da sequência de Fibonacci e armazenar o resultado no registrador **R5**.

H) Exporte o trabalho realizado num arquivo zip: ao final de cada atividade prática, será exigida a entrega de um programa na forma de um arquivo **zip** contendo o seu projeto CCS. Para exportar o seu trabalho, clique em "**File**" -> "**Export**" e selecione a opção **"Archive File"** dentro da pasta "**General**".