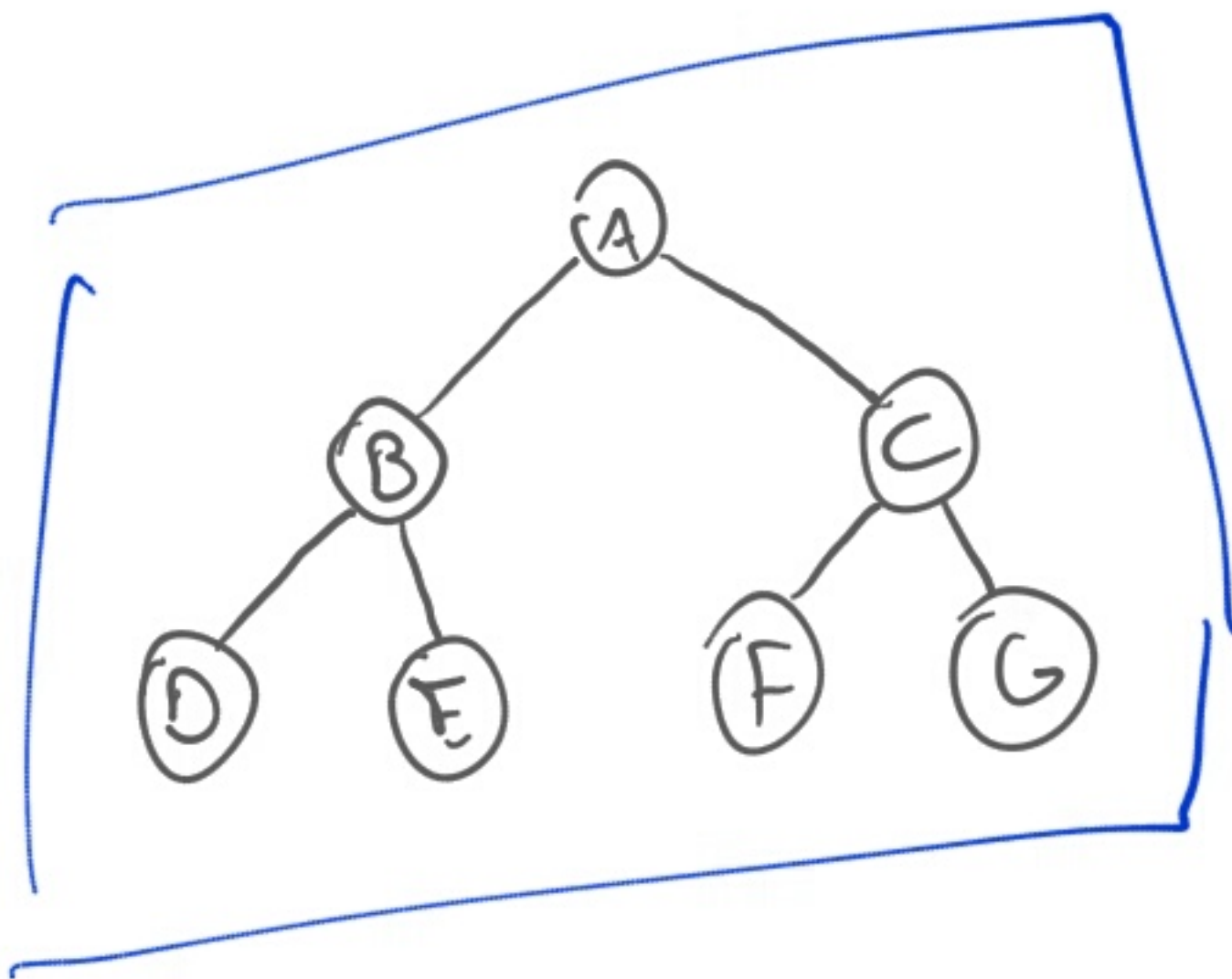


```
1;-----
2; MSP430 Assembler Code Template for use with TI Code Composer Studio
3;
4;
5;-----
6      .cdecls C,LIST,"msp430.h"      ; Include device header file
7
8;-----
9      .def      RESET                ; Export program entry-point to
10                                     ; make it known to linker.
11;-----
12      .text                          ; Assemble into program memory.
13      .retain                        ; Override ELF conditional linking
14                                     ; and retain current section.
15      .retainrefs                    ; And retain any sections that have
16                                     ; references to current section.
17
18;-----
19 RESET      mov.w    #_STACK_END,SP    ; Initialize stackpointer
20 StopWDT     mov.w    #WDTPW|WDTHOLD,&WDCTL ; Stop watchdog timer
21
22
23;-----
24;Main loop here
25;Daniel Moraes da Silva - Engenharia de Computação
26;Percurso Arvore: Pré-Ordem
27;-----
28 PO:        mov.b    #arv,R5
29            mov.b    #ARV,R9
30            mov.b    #0,R6            ;inidice inicia com a Raiz
31            mov.b    #3,R7            ;indice da subarvore da direita
32            mov.b    #6,R8            ;indice da subarvore da esquerda
33
34 NOD:        add.b    #1,R6            ;andando um indice do vetor
35            mov.b    arv(R6),ARV(R6) ;conferindo o atual indice colocando o elemento em R4
36            cmp.b    R7,R6            ;comparar indices
37            jhs      NOE              ;se o atual indice for maior que subarvore da Direita (R7), Ir para subarvore Esquerda
38            call     #NOD              ;se não continua percorrendo a subarvore da Direita
39
40 NOE:        add.b    #1,R6            ;andando um indice do vetor
41            mov.b    arv(R6),ARV(R6) ;conferindo o atual indice colocando o elemento em R4
42            cmp.b    R8,R6            ;comparar indices
43            jhs      FIM              ;se o atual indice for maior que subarvore da Esquerda (R8), já percorreu toda a Arvore, chama F
44            call     #NOE              ;se não continua percorrendo a subarvore da Direita
45
46
47 FIM:        jmp      $
48
49
50            .data
51 arv:         .byte    "ABCDEFG"
52
53            .data
54 ARV:         .byte    "000000"
55
56;-----
57; Stack Pointer definition
58;-----
59            .global  __STACK_END
60            .sect    .stack
61
62;-----
63; Interrupt Vectors
64;-----
65            .sect    ".reset"          ; MSP430 RESET Vector
66            .short   RESET
67
68
```

Pré - Ordem



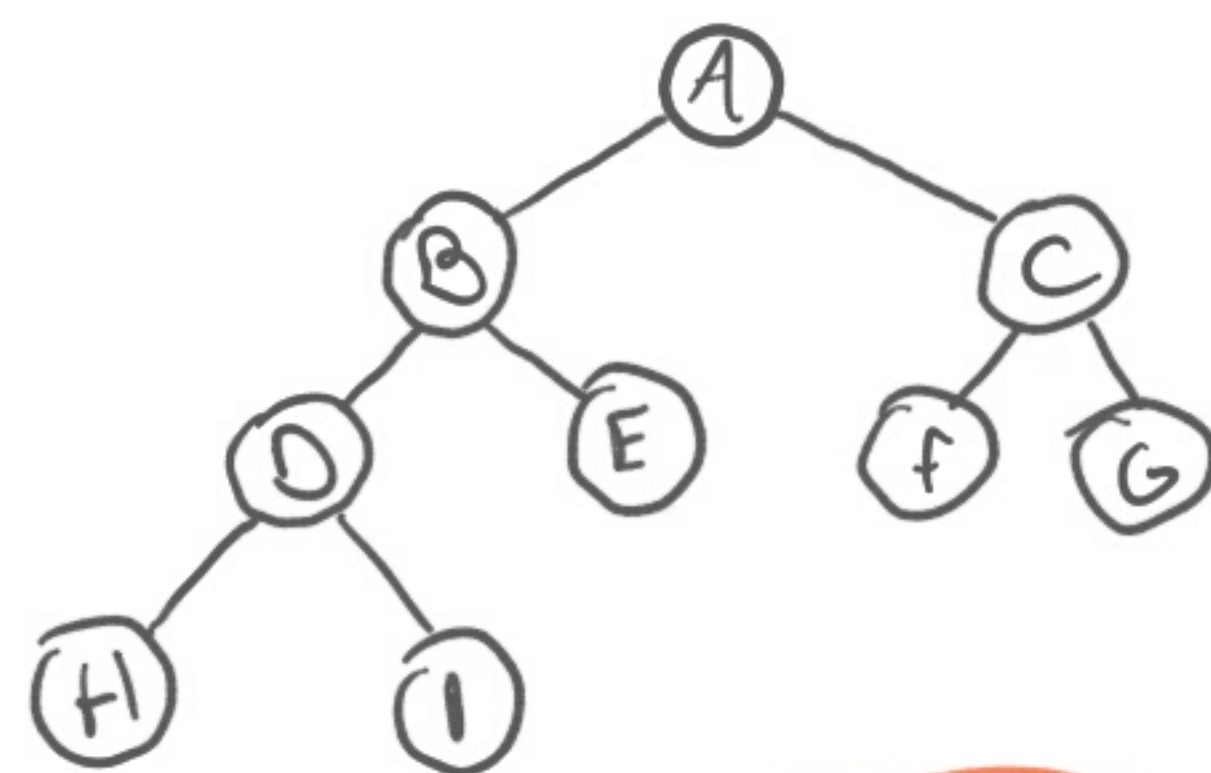


```

1;-----
2; MSP430 Assembler Code Template for use with TI Code Composer Studio
3;
4;
5;-----
6      .cdecls C,LIST,"msp430.h"      ; Include device header file
7
8;-----
9      .def      RESET                  ; Export program entry-point to
10                                         ; make it known to linker.
11;-----
12      .text                          ; Assemble into program memory.
13      .retain                        ; Override ELF conditional linking
14                                         ; and retain current section.
15      .retainrefs                    ; And retain any sections that have
16                                         ; references to current section.
17
18;-----
19 RESET      mov.w    #__STACK_END,SP    ; Initialize stackpointer
20 StopWDT     mov.w    #WDTPW|WDTHOLD,&WDTCTL ; Stop watchdog timer
21
22
23;-----
24; Main loop here
25; Daniel Moraes da Silva - Engenharia de Computação
26; Percurso Arvore: Ordem Simetrica
27;-----
28 OS:        mov.b    #arv,R4
29             mov.b    #ARV,R5
30             mov.b    #0,R6
31
32 SUBESQ:     mov.b    #7,R7             ;|
33             mov.b    arv(R7),ARV(R6)  ;|> Imprime H
34             add.b    #1,R6             ;|
35
36             mov.b    #3,R7             ;|
37             mov.b    arv(R7),ARV(R6)  ;|> Imprime D
38             add.b    #1,R6             ;|
39
40             mov.b    #8,R7             ;|
41             mov.b    arv(R7),ARV(R6)  ;|> Imprime I
42             add.b    #1,R6             ;|
43
44             mov.b    #1,R7             ;|
45             mov.b    arv(R7),ARV(R6)  ;|> Imprime B
46             add.b    #1,R6             ;|
47
48             mov.b    #4,R7             ;|
49             mov.b    arv(R7),ARV(R6)  ;|> Imprime E
50             add.b    #1,R6             ;|
51
52             mov.b    #0,R7             ;|
53             mov.b    arv(R7),ARV(R6)  ;|> Imprime A
54             add.b    #1,R6             ;|
55             call     #SUBDIR           ;chama subrotina para percorrer a subArvore da a direita
56
57 SUBDIR:     mov.b    #5,R7             ;|> Imprime F
58             mov.b    arv(R7),ARV(R6)  ;|
59             add.b    #1,R6             ;|
60
61             mov.b    #2,R7             ;|> Imprime C
62             mov.b    arv(R7),ARV(R6)  ;|
63             add.b    #1,R6             ;|
64
65             mov.b    #6,R7             ;|> Imprime G
66             mov.b    arv(R7),ARV(R6)  ;|
67             add.b    #1,R6             ;|
68             call     #FIM
69
70 FIM:        jmp      $
71
72
73             .data
74 arv:         .byte    "ABCDEFGHI"
75
76             .data
77 ARV:         .byte    "00000000"
78
79;-----
80; Stack Pointer definition
81;-----
82             .global  __STACK_END
83             .sect    .stack
84
85;-----
86; Interrupt Vectors
87;-----
88             .sect    ".reset"          ; MSP430 RESET Vector
89             .short   RESET
90

```

Ordem Simétrica 1



H, D, I, B, E, A, F, C, G

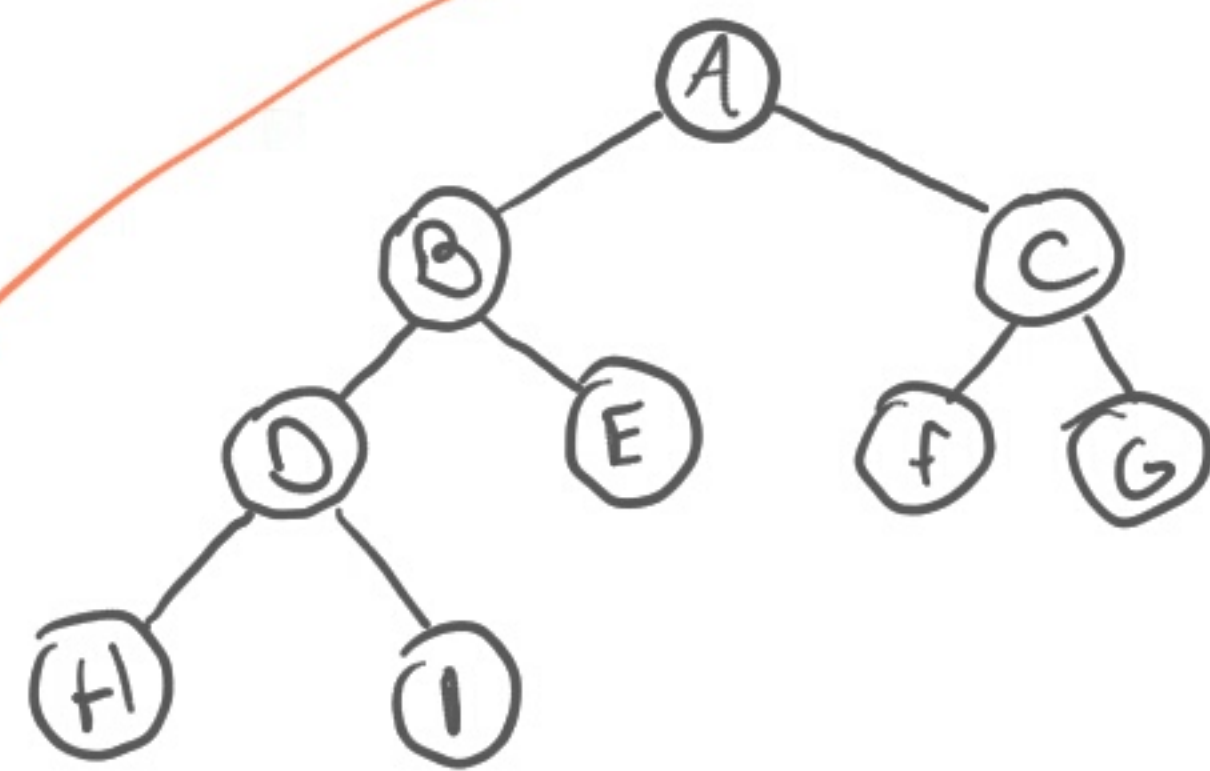


```

1;-----
2; MSP430 Assembler Code Template for use with TI Code Composer Studio
3;
4;
5;-----
6      .cdecls C,LIST,"msp430.h"      ; Include device header file
7
8;-----
9      .def      RESET                ; Export program entry-point to
10                                     ; make it known to linker.
11;-----
12      .text                          ; Assemble into program memory.
13      .retain                        ; Override ELF conditional linking
14                                     ; and retain current section.
15      .retainrefs                    ; And retain any sections that have
16                                     ; references to current section.
17
18;-----
19 RESET      mov.w    #_STACK_END,SP    ; Initialize stackpointer
20 StopWDT    mov.w    #WDTPW|WDTHOLD,&WDTCTL ; Stop watchdog timer
21
22
23;-----
24 ;Main loop here
25 ;Daniel Moraes da Silva - Engenharia de Computação
26 ;Percurso Arvore: Ordem Final
27;-----
28 OS:        mov.b    #arv,R4
29            mov.b    #ARV,R5
30            mov.b    #0,R6
31
32 SUBESQ:    mov.b    #7,R7              ;|
33            mov.b    arv(R7),ARV(R6)    ;|> Imprime H
34            add.b    #1,R6              ;|
35
36            mov.b    #8,R7              ;|
37            mov.b    arv(R7),ARV(R6)    ;|> Imprime I
38            add.b    #1,R6              ;|
39
40            mov.b    #3,R7              ;|
41            mov.b    arv(R7),ARV(R6)    ;|> Imprime D
42            add.b    #1,R6              ;|
43
44            mov.b    #4,R7              ;|
45            mov.b    arv(R7),ARV(R6)    ;|> Imprime E
46            add.b    #1,R6              ;|
47
48            mov.b    #1,R7              ;|
49            mov.b    arv(R7),ARV(R6)    ;|> Imprime B
50            add.b    #1,R6
51            call     #SUBDIR            ;chama subrotina para percorrer a subArvore da a direita
52
53 SUBDIR:    mov.b    #5,R7              ;|
54            mov.b    arv(R7),ARV(R6)    ;|> Imprime F
55            add.b    #1,R6
56
57            mov.b    #6,R7              ;|
58            mov.b    arv(R7),ARV(R6)    ;|> Imprime G
59            add.b    #1,R6
60
61            mov.b    #2,R7              ;|
62            mov.b    arv(R7),ARV(R6)    ;|> Imprime C
63            add.b    #1,R6
64
65            mov.b    #0,R7              ;|
66            mov.b    arv(R7),ARV(R6)    ;|> Imprime A
67            add.b    #1,R6
68            call     #FIM
69
70
71 FIM:      jmp      $
72
73
74      .data
75 arv:      .byte    "ABCDEFGHI"
76
77      .data
78 ARV:      .byte    "00000000"
79
80
81;-----
82 ; Stack Pointer definition
83;-----
84      .global __STACK_END
85      .sect    .stack
86
87;-----
88 ; Interrupt Vectors
89;-----
90      .sect    ".reset"                ; MSP430 RESET Vector
91      .short   RESET
92
93

```

Ordem final ↗



H, I, D, E, B, F, G, C, A