## RAMSES - Principais Características

• Largura de dados e de endereços de 8 bits
• Dados representados em complemento de dois
• 4 modos de endereçamento: direto, indireto, imediato e indexado
• 2 registradores de uso geral de 8 bits (B e X)
• 1 registrador de índice de 8 bits
• 1 apontador de programa de 8 bits (PC)
• 1 registrador de estado com 3 códigos de condição: negativo, zero e carry – sendo que o carry também atua como um borrow (RA)

**MODOS DE INSTRUÇÃO**

**DIRETO** - Sintaxe: a - O valor do argumento 'a' representa o endereço onde o operando se encontra ou, em operações de jump, o endereço para qual o desvio será realizado.

**INDIRETO** - Sintaxe: a,I (valor com sufixo ,I) - O valor do argumento 'a' representa um endereço que por sua vez contém o endereço direto.

**IMEDIATO** - Sintaxe: #a (valor com prefixo #) - O valor do argumento 'a' não representa um endereço, e sim o valor imediato que deve ser carregado no registrador (instrução LDR) ou utilizado em operações aritméticas.

**INDEXADO POR X** - Sintaxe: a,X (valor com sufixo ,x - Endereçamento direto com deslocamento. A soma dos valores de 'a' e do registrador X representa o endereço direto.

**Instruções**

•**Store Register**: STR r a - Armazena o valor do registrador 'r' no endereço 'a'.
•**Load Register**: LDR r a - Carrega o valor no endereço 'a' para o registrador 'r'
•**ADD:** ADD r a - Adiciona o valor no endereço 'a' ao registrador 'r'
•**OR**: OR r a - Realiza um 'ou' lógico entre cada bit de 'a' e o bit correspondente no registrador 'r'.
•**AND** r a - Realiza um 'e' lógico entre cada bit de 'a' e o bit correspondente no registrador 'r'
•**NOT**: NOT r - Inverte (complementa) o valor dos bits do registrador 'r'
•**SUBTRACT**: SUB r a - Subtrai o valor no endereço 'a' do registrador 'r'
•**Jump on Carry**: JC a - Se a flag C estiver ativada (carry), desvia a execução para o endereço 'a'
•**Negate**: NEG r - Troca o sinal do valor em complemento de 2 do registrador 'r' de positivo para negativo e vice-versa
•**Shift Right**: SHR r - Realiza shift lógico dos bits do registrador 'r' para a direita, passando o estado do bit menos significativo para a flag C (carry) e preenchendo o bit mais significativo com 0

----------///----------

## CESAR16 – Principais Características

• Processamento de pilha / comp de 2 / 16 bits
• 8 modos de endereçamento nativos + 4 modos de endereçamento derivados
• 8 registradores de 16 bits (R0, R1, R2, R3, R4, R5, R6 e R7)
• 6 registradores de uso geral
• 1 apontador de programa (R7 – PC - program counter)
• 1 apontador de pilha (R6 – SP - stack pointer)
• 1 registrador de estado com 4 códigos de condição: negativo, zero, carry e overflow
• 2 periféricos: teclado e visor de **26** caracteres
• TIMDT - TIMer DaTa: um byte usado para configurar a periodicidade de interrupção do timer, em milisegundos
• INTS - INTerrupt Status: um byte que informa qual o periférico solicitou a interrupção
• INTE - INTerrupt Enable: um byte que é usado para controlar, em geral, a habilitação das interrupções e para habilitar cada uma delas individualmente
• IVET - Interrupt VEcTor: dois bytes onde deve-se escrever o endereço da ISR - Interrupt Service Routine.

•**Exemplo de uma escrita no display:**

```
; Área Reservada
; Acesso em 16 bits
                org             hff80
STACK:          daw             [31 ; Área reservada
IVET:   dw              0       ; Vetor de interrupção
; Acesso em 8 bits
                dab             [24]
INTS:   db 0;  INTERRUPT STATUS: IP x x x . x x IPStec IPStim
INTE:   db 0; INTERRUPT ENABLE: IE x x x . x x IEStec IEStim
TECST:  db              0       ; Status do teclado
TECDT:  db              0       ; Dado do teclado
VISOR:  dab             [36]    ; Portas de acesso ao visor
                org             0
                mov             #STACK,r6
                mov             #MSG1,r0
                mov             #17,r1
                jsr             r7,Display1
                hlt
MSG1:           dab             'Mensagem Numero 1'
; void Display1( r0 = Endereço de inicio da mensagem; r1 = Tamanho da mensagem)
Display1:       mov             #VISOR,R2
                dec             r0
Display1_1:     mov             (r0),(R2)
                inc             r0
                inc             r2
                dec             r1
                bne             Display1_1
                rts             r7
==========FIM DO PROGRAMA==========
```

•**Programa tecbuf**

- Programa para comparar o comportamento da leitura de teclado de forma direta e usando a interrupção

```
;O programa verifica se tem tecla.
; Se tiver, colocar no visor;
; Se não tiver, entra em um wait de 10 segundos
; No laco principal tem duas funções:
; DisplayTeclado: lê o teclado e coloca no visor
; Wait: fornece a temporização de 10 segundos
; São duas versões do programa
;1. Versão sem interrupção -> mostra-se que serão perdidas teclas, mesmo com um buffer no PP
;2. Versão com interrupção -> mostra-se que as teclas podem ser armazenadas em um buffer, na ISR
; Versão 1
;<<1>> Configuração da interrupção
;<<2>> Configuração da leitura de teclado
;Acesso em 16 bits
                org             hff80
STACK:
                daw             [31]            ; Área reservada
IVET:   dw              0                       ; Vetor de interrupção
; Acesso em 8 bits
                dab             [24]
INTS:   db              0                       ; INTERRUPT STATUS: IP x x x . x x IPStec IPStim
INTE:   db              0                       ; INTERRUPT ENABLE: IE x x x . x x IEStec IEStim
TECST:  db              0                       ; Status do teclado
TECDT:  db              0                       ; Dado do teclado
VISOR:  dab             [36]            ; Portas de acesso ao visor
org             0 ; Inicializa o processador e o hardware
                mov             #STACK,r6
```

```
                clr             INTS; Reset pedidos de interrupç.
                clr     TECST   ; Reset estado do teclado
                mov             #isr,IVET
; Set Interrupt Service Rotine
; Inicializa variaveis de operação das rotinas do programa
                mov             #VISOR,POSICAO          ;
POSICAO = VISOR
                jsr             r7,ClearDisplay
; Programa principal

; <<1>> Configuração da interrupção
                mov             #h00,INTE               ;
Sem interrupção
;               mov             #h82,INTE               ;
Com interrupção de teclado - Habilita as interrupções necessárias:
TECLADO=0x82   TIMER=0x81
main:
                jsr             r7,DisplayTeclado
                jsr             r7,Wait
                jmp             main
; Le teclado e coloca no visor
DisplayTeclado:
                jsr             r7,TeclaDisponivel
                tst             r0
                beq             DT_Fim
                jsr             r7,GetTecla
                cmp             r0,#' '
                blt             DT_Fim
                cmp             r0,#'z'
                bgt             DT_Fim
                mov             POSICAO,r1
                mov             r0,(r1)
                inc             r1
                bne             CP_Inc
                mov             #VISOR,r1
        ;POSICAO = VISOR
CP_Inc:
                mov             r1,POSICAO
                jmp             DisplayTeclado
DT_Fim:
                rts             r7
; Simula um processamento demorado
Wait:
                mov             #10,r0
WaitLoopR0:
                mov             #30000,r1
WaitLoopR1:
                sob             r1,WaitLoopR1
                sob             r0,WaitLoopR0
                rts             r7
; Limpa o visor
ClearDisplay:
                mov             #VISOR,r0
CD_Loop:
                mov             #' ',(r0)
                inc             r0
                bne             CD_Loop
                rts             r7
; TeclaDisponivel   -> Rotina que informa se algo foi digitado
; GetTecla          -> Rotina que retorna a tecla digitada
; <<2>> Configuração da leitura de teclado
; Rotina que informa se algo foi digitado
; Retorna R0==0, se nada digitado; R0!=0, se algo digitado
TeclaDisponivel:    jmp             TeclaDisponivel_1   ; -> leitura
direta
;TeclaDisponivel:   jmp             TeclaDisponivel_2   ; -> leitura por
interrupção
; Rotina que retorna a tecla digitada
GetTecla:                   jmp             GetTecla_1
;GetTecla:                  jmp             GetTecla_2
; Versão 1 das funções
TeclaDisponivel_1:
                mov             TECST,r0
                rts             r7
GetTecla_1:
                mov             TECDT,r0
                clr             TECST
                rts             r7
; Versão 2 das funções
TeclaDisponivel_2:  and         #h7f,INTE           ; IE = 0;
                clr             r0
                ; R0 = (prOut==ptIn ? FALSE:TRUE);
                cmp             ptOut, ptIn
                beq             TD_2_1
                mov             #1,r0
TD_2_1:         or              #h80,INTE           ; IE = 1;
                rts             r7
GetTecla_2:     and             #h7f,INTE           ; IE = 0
                mov             ptOut,r1    ; r0 = *ptOut++
                dec             r1
                mov             TEC_BUFFER(r1),r0
                add             #2,r1
                and             #hff,r0     ; r0 &= 0xFF
                and             #h1f,r1
            ; ptOut = ptOut & 0x1F // loop around pointer
                mov             r1,ptOut
                or              #h80,INTE           ; IE = 1
                rts             r7
; Rotina de INTERRUPÇÃO
; Só será chamada se as interrupções estiverem habilitadas
isr:
                mov             r0,-(r6)    ; Salva registradores
                mov             r1,-(r6)
                mov             r2,-(r6)
                mov             r3,-(r6)
                mov             r4,-(r6)
                mov             r5,-(r6)
                mov             INTS,r0     ; Verifica se é INT do
TECLADO
                and             #2,r0
                beq             ISR3
                jsr             r7,ISRtec   ; Tratamento da INT do
TECLADO
                and             #hFFFD,INTS         ; Desliga bit de
INT TECLADO
ISR3:
                mov             (r6)+,r5    ; Retorna os
registradores da pilha
                mov             (r6)+,r4
                mov             (r6)+,r3
                mov             (r6)+,r2
                mov             (r6)+,r1
                mov             (r6)+,r0
                rti
; Tratamento das interrupções de teclado
ISRtec:
                tst             TECST; if (Tecla está disponivel?) {
                beq             ISRtec1
                mov             ptIn,r2     ; p = (in+1)&0x1F
                inc             r2
                and             #h1F,r2
                cmp             r2,ptOut                ;
                if (p!=out) { // Verifica se o BUFFER ESTA CHEIO
                beq             ISRtec_2
                mov             ptIn,r1     ;*in = TECLA
                dec             r1
                mov             TEC_BUFFER(r1),r0
                and             #hFF00,r0
                or              TECDT,r0
                mov             r0,TEC_BUFFER(r1)
                mov             r2,ptIn     ;in = p
ISRtec_2:
                clr     TECST
ISRtec1:
                rts     r7
; Variaveis do programa
; Ocupam a memória logo após o final do programa
;
POSICAO:dw              0               ; Endereço no visor da
posição atual onde o 'A' está aparecendo
```