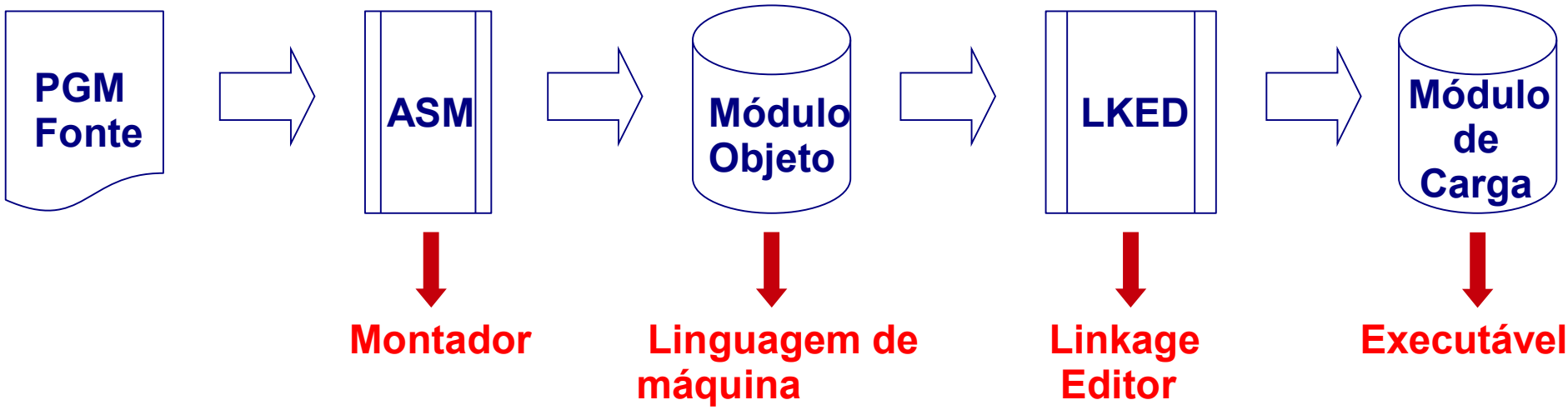
ASSEMBLER BÁSICO

Montador Assembler



PROCEDURES: (SYS1.PROCLIB)

ASMCL – Compila e Linkedita

ASMCLG – Compila Linkedita e Executa

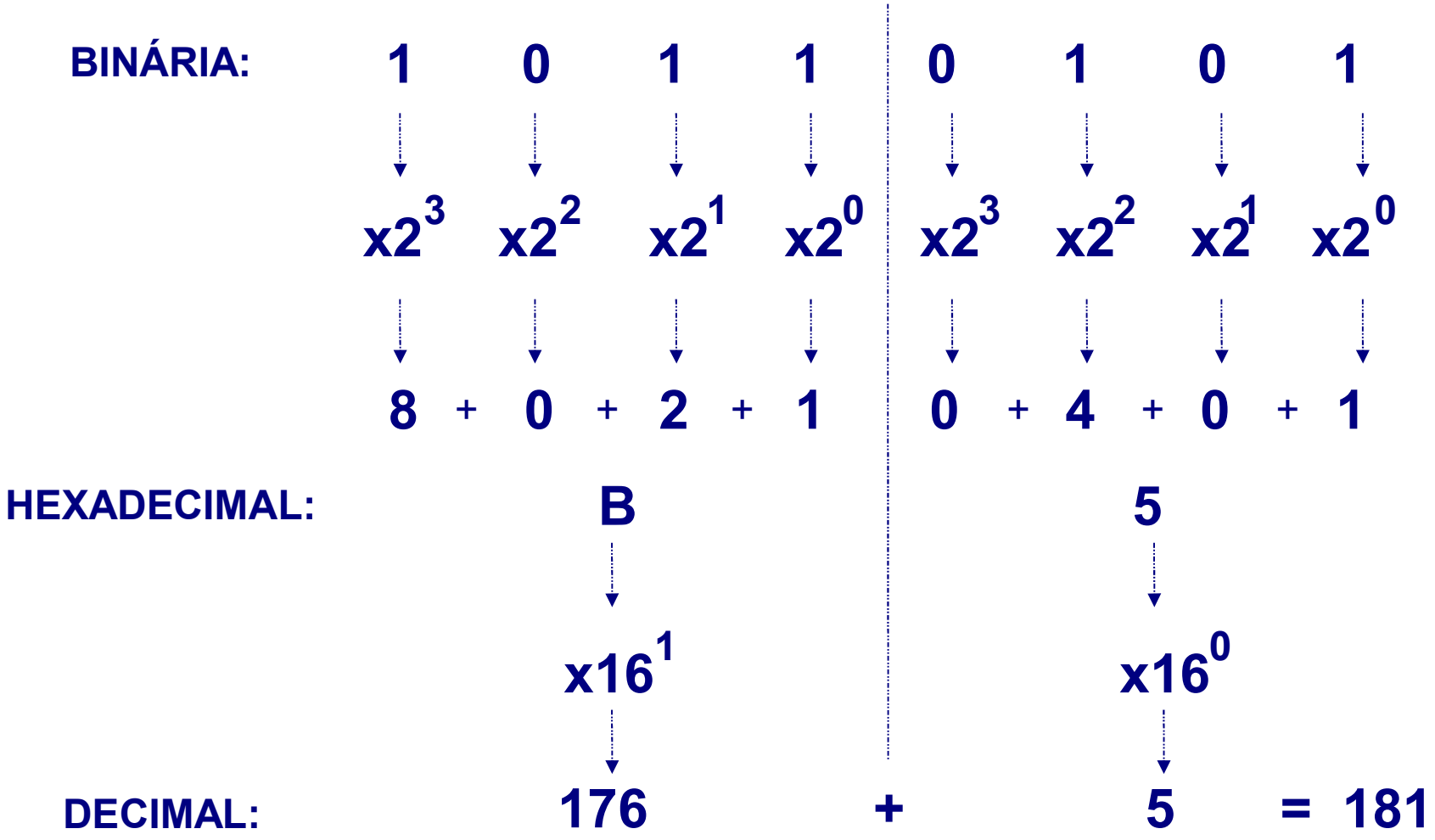
Sistemas de Numeração

BINÁRIA x DECIMAL

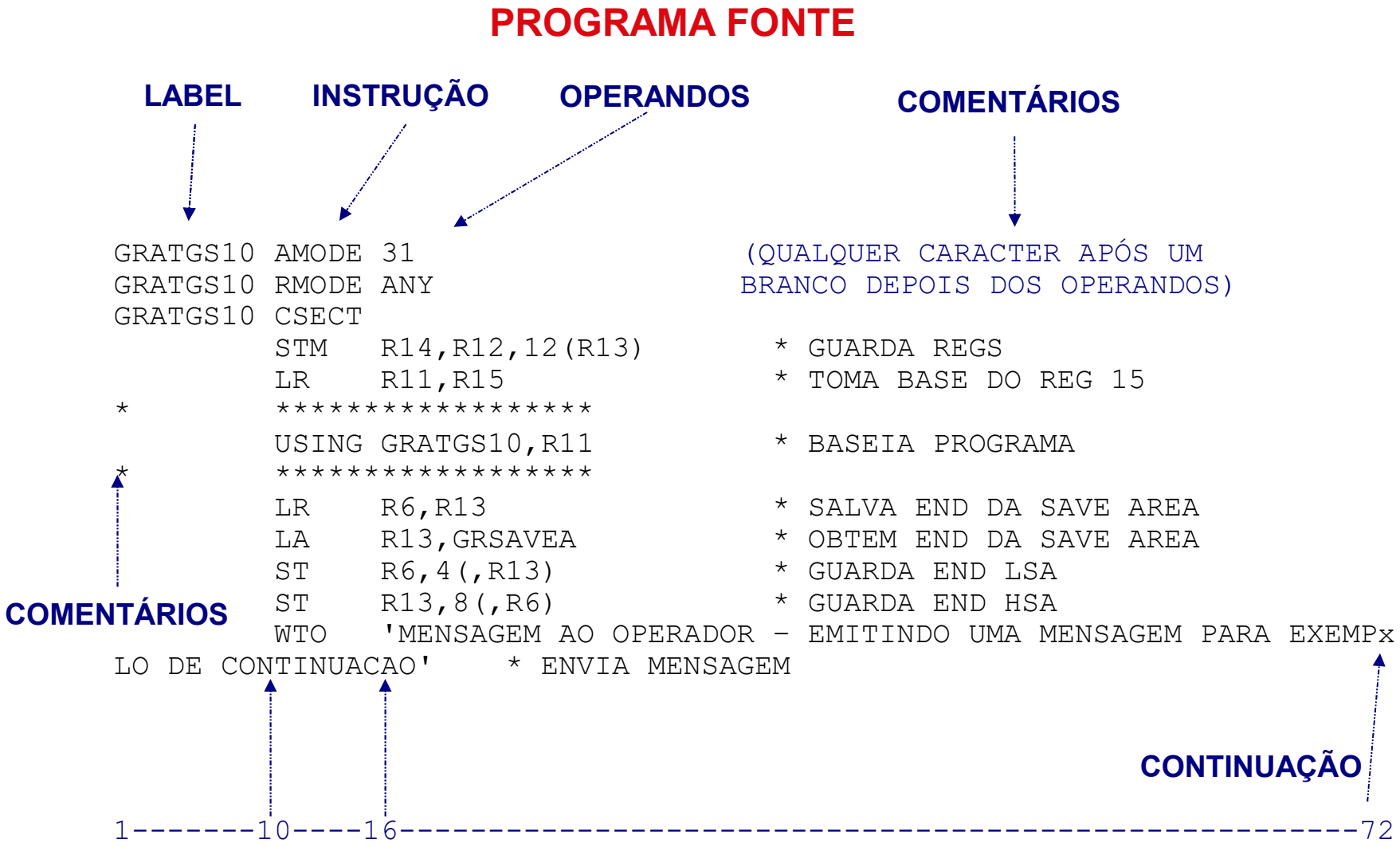
BINÁRIA:	1	0	1	1	0	1	0	1
	↓	↓	↓	↓	↓	↓	↓	↓
	$x2^7$	$x2^6$	$x2^5$	$x2^4$	$x2^3$	$x2^2$	$x2^1$	$x2^0$
	↓	↓	↓	↓	↓	↓	↓	↓
DECIMAL:	128 +	0 +	32 +	16 +	0 +	4 +	0 +	1 = 181

Sistemas de Numeração

BINÁRIA x HEXADECIMAL x DECIMAL



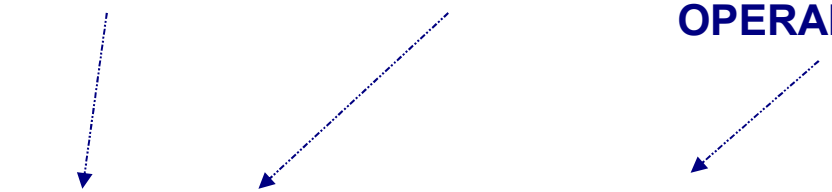
Estrutura de um programa Assembler



Estrutura de um programa Assembler

PROGRAMA COMPILADO

ENDEREÇO **INSTRUÇÃO** **ENDEREÇOS OPERANDOS**



```
  Loc  Object Code    Addr1 Addr2  Stmt   Source Statement
                                       1 CICPGS10 AMODE 31
                                       2 CICPGS10 RMODE ANY
000000                 00000 004CF     3 CICPGS10 CSECT
                                       4          PRINT NOGEN
000000 90EC D00C             0000C     6          STM   R14,R12,12(R13)
000004 18BF                            7          LR    R11,R15
                                       8 *        ******************
                 R:B   00000           9          USING CICPGS10,R11
                                      10 *        ******************
000006 186D                           11          LR    R6,R13
000008 41D0 B38C             0038C    12          LA    R13,CISAVEA
00000C 5060 D004             00004    13          ST    R6,4(,R13)
000010 50D0 6008             00008    14          ST    R13,8(,R6)
```

Endereçamento

- **AMODE** – Addressing Mode
  - Formato dos endereços tratados pelo programa:
    - **AMODE 24** – Programas com endereços de 24 bits – 16 MB
    - **AMODE 31** – Programas com endereços de 31 bits – 2 GB
    - **AMODE 64** – Programas com endereços de 64 bits – 16 EB

- **RMODE** – Residence Mode
  - Localização do programa na memória virtual:
    - **RMODE 24** – Programa reside abaixo da linha dos 16 MB
    - **RMODE 31** ou **RMODE ANY** – Programa pode residir acima ou abaixo da linha dos 16 MB
    - **RMODE 64** – Programa reside acima da barra dos 2 GB

Endereçamento

Registradores Gerais

- Conjunto de 16 elementos de acesso muito mais rápido do que à memória comum
- Usados para endereçar dados (registradores base e de índice), como acumuladores ou com funções específicas para algumas instruções
- No Z/OS os registradores gerais têm 64 bits
- São numerados de 0 a 15
- Os registradores 0,1,13,14 e 15 são utilizados por macros e/ou em Linkage Convention (cuidado ao utilizá-los no programa)
- Algumas instruções usam pares de registradores

Endereçamento

Endereço de Memória

- Para localizar um endereço de memória necessitamos de um registrador base e um deslocamento (e, eventualmente, de um indexador). Para obter o endereço efetivo, o hardware soma os componentes.
- O registrador 0 não pode ser usado nem como base nem como indexador.
- Nas instruções de máquina o deslocamento tem sempre 12 bits, portanto, o deslocamento pode variar de 0 a 4095 (x'000' a x'FFF')
- Se o programa for maior do que 4K é necessário definir mais de um registrador base ou dividi-lo em várias sessões (CSECT)
- O registrador de índice pode ter um conteúdo positivo ou negativo

Endereçamento

Demais Registradores

- **Registradores de ponto flutuante** – Usados para cálculos com números não inteiros, fracionários ou com expoentes.
- **Registradores de controle** – Funções específicas utilizadas pelo programa supervisor.
- **Registradores de acesso** – Para acesso a dados em outros address spaces ou data spaces.

Endereçamento

WORDS (palavras)

- Todo byte tem um endereço na memória. Juntando-se os bytes formamos as palavras:
  - **HALFWORD** - 2 bytes
  - **FULLWORD** - 4 bytes
  - **DOUBLEWORD** - 8 bytes
  - **QUADWORD** - 16 bytes

  Endereços alinhados em **HALFWORD** são múltiplos de 2, **FULLWORD** múltiplos de 4 e **DOUBLEWORD** múltiplos de 8.

Constantes e Storage

- **DC – Define Constant**
  - Cria e inicializa os bytes de uma área de memória com um conteúdo definido.
  - Exemplo: Definir uma constante chamada ‘CEM’ com o valor ‘100’

    CEM DC C’100’

- **DS – Define Storage**
  - Só reserva espaço na memória. O conteúdo é indefinido.
  - Exemplo: Reservar uma área de memória de ’50’ bytes com o nome de ‘ENTRADA’

    ENTRADA DS CL50

Constantes e Storage

- **DC e DS** – Fator de duplicação e tamanho do campo

  Podemos definir várias constantes ou storage numa única instrução utilizando o fator de duplicação e/ou o de tamanho de campo.

  - Exemplo1: Definir uma constante de **100** bytes com o conteúdo **'A'**
    TABELA DC 100C'A' (onde **100** é o fator de duplicação)

  - Exemplo2: Reservar uma área de memória de **100** bytes
    TABELA DS CL100 (onde **L** indica o tamanho do campo)

- **DC e DS** – Tipo do campo

  Mais usados:

B – binário	**C** – character
X – hexadecimal	**P** – compactada
A – endereço interno	**V** – endereço externo

  Exemplo: Definir uma constante de **1** byte chamada **HEXA01** com o conteúdo hexadecimal **'7F'.**

  HEXA01 DC X'7F' (onde **X** é o tipo de campo)

Literais

- São constantes que não têm um nome, ou seja, são definidas pela instrução que a utiliza
- Todas as literais são organizadas pelo assembler após a instrução LTORG
- Uma literal não pode ser alterada diretamente pelo programa
- Literais iniciam pelo caracter ‘=‘ na instrução que a define
- Exemplo: Comparar o conteúdo do registrador **1** com **’30’**

  C 1,=F’30’

Instruções ao Montador

CSECT e DSECT

- **CSECT –** Control Section
  - Inicia ou continua uma sessão do programa.
  - Programas pequenos têm somente uma CSECT
  - Programas maiores são subdivididos em várias CSECT
  - Exemplo: Definir uma CSECT inicial de um programa chamado CICP00
    CICP00 CSECT

- **DSECT** – Dummy Section
  - Mapeia áreas da memória atribuindo nomes aos campos
  - Não é gerado nenhum código objeto pelo compilador
  - Exemplo: Descrever os campos de uma tabela carregada
    TAB DSECT

Instruções ao Montador

USING e DROP

- **USING**
  - Define um endereço base para uma CSECT ou DSECT
  - Um registrador deve apontar para a área que está sendo baseada
  - Exemplo: Definir um endereço base para uma CSECT CIC

    USING CIC,12 (onde **12** é o registrador)

- **DROP**
  - Encerra o endereçamento base estabelecido pela USING
  - Utilizado para liberar um registrador para basear uma outra CSECT ou DSECT
  - Exemplo: Liberar o registrador base da CSECT CIC

    DROP 12

Instruções ao Montador

START e END

- **START** – Início da primeira CSECT do programa
  - Define a primeira instrução do programa
  - Se não informada CSECT, será criada um a partir dele
  - Exemplo: Iniciar uma CSECT chamada **SIAD**

    SIAD START

- **END** – Fim do programa
  - Encerra o processo de compilação do programa
  - Deve ser a última instrução informada no programa fonte
  - Exemplo: Encerrar a compilação do programa **CICP01**

    END CICP01

Instruções ao Montador

EQU e LTORG

- **EQU** – Equate
  - Define um valor para um símbolo
  - Exemplo: Definir um campo chamado RETCODE4 cujo valor é ‘4’

    RETCODE4 EQU C’4’

    EQU * (* indica que o valor do equate é o do endereço atual)

- **LTORG** – Literal organizaion
  - Define a área onde serão gerados, pelo compilador assembler, as literais definidas no programa
  - Se **LTORG** for omitido, as literais são criadas após a instrução **END**
  - Um programa com várias **CSECT** pode ter um **LTORG** para cada uma

Formato das Instruções

- **Instruções**
  - As instruções podem ter **2**, **4** ou **6** bytes de tamanho
  - As instruções devem estar alinhadas por **HALFWORD**
  - O primeiro byte ou os 2 primeiros bytes indicam o código (hexadecimal) da instrução a ser executada
  - Os 2 primeiros bits do código da instrução indicam o seu tamanho:

Bits 0 e 1	**Tamanho da Instrução**
00	2 bytes
01	4 bytes
10	4 bytes
11	6 bytes

Formato das Instruções

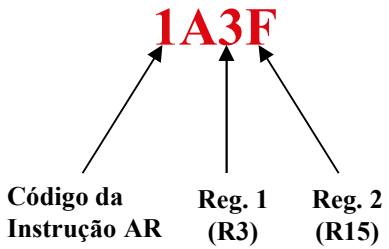
- **Formato RR – Registrador e Registrador**

  Tamanho – 2 bytes

  Exemplo: Somar os registradores R3 e R5

  AR 3,5

  Código de máquina gerado:



Formato das Instruções

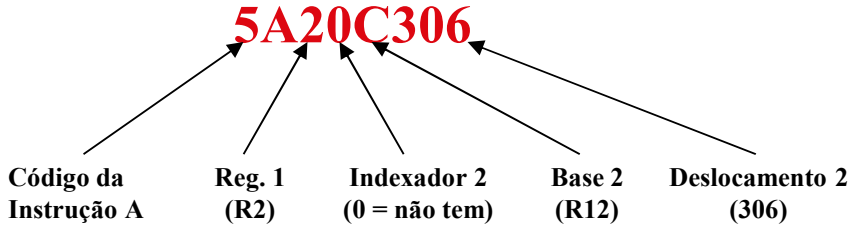
- **Formato RX – Registrador e memória com indexador**

  Tamanho – 4 bytes

Exemplo: Somar o registrador R2 com o conteúdo do campo CONTADOR

A 2,CONTADOR

Código de máquina gerado: (supondo R12 como base do programa e ‘306’ como deslocamento do campo CONTADOR)



Formato das Instruções

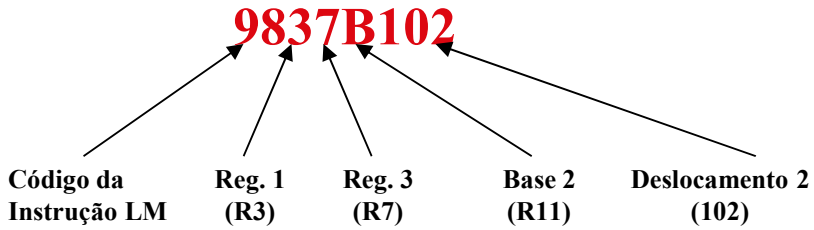
- **Formato RS – Registrador e memória**

  Tamanho – 4 bytes

Exemplo: Carregar os registradores 3 a 7 a partir do campo SAVEREGS

LM 3,7,SAVEREGS

Código de máquina gerado: (supondo R11 como base do programa e '102' como deslocamento do campo SAVEREGS)



Formato das Instruções

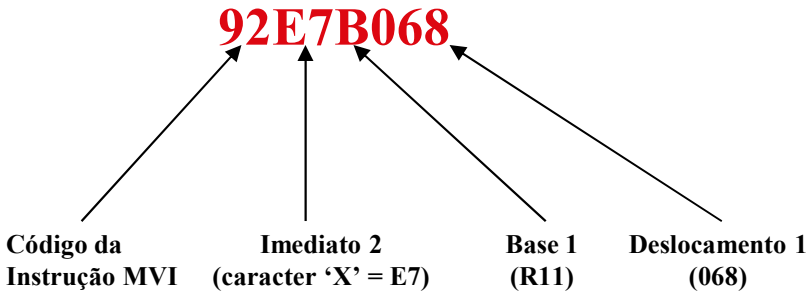
- **Formato SI – Memória e imediato**

  Tamanho – 4 bytes

Exemplo: Mover o caracter ‘X’ para o campo BCT102

MVI BCT102,C’X’

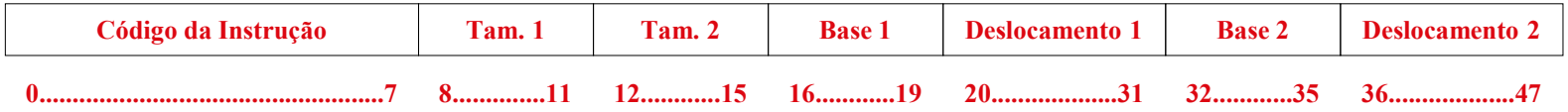
Código de máquina gerado: (supondo R11 como base do programa e ‘068’ como deslocamento do campo BCT102)



Formato das Instruções

- **Formato SS – Memória e memória**

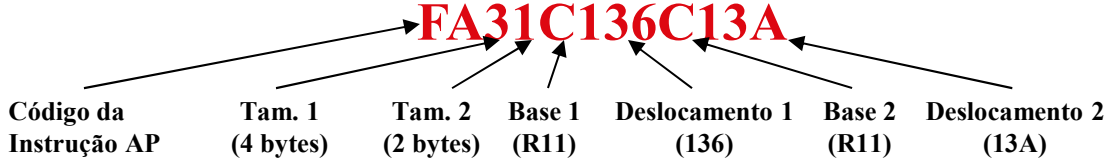
  Tamanho – 6 bytes



Exemplo: Somar os dois campos decimais compactados TOTAL e CAMPO1

AP TOTAL,CAMPO1

Código de máquina gerado: (supondo R11 como base do programa, '136' e '13A' os deslocamentos dos campos TOTAL e CAMPO1, 2 bytes de tamanho do CAMPO1 e 4 bytes o tamanho do TOTAL)



Formato das Instruções

Outros formatos

RRE – Registrador e registrador com código da instrução estendido

RRF – Registrador e registrador com código da instrução estendido e campos adicionais

RXE – Registrador e memória com indexador e código da instrução estendido

RXF – Registrador e memória com indexador e código da instrução estendido

RSL – Memória com código da instrução estendido

RIE – Registrador e imediato com código da instrução estendido

RIL – Registrador e imediato longo com código da instrução estendido

Início e término de um programa

Registrador Base

Ao receber o controle do Sistema Operacional, o endereço do programa que está iniciando vem no registrador geral R15.

O programa chamado deve esse endereço no registrador escolhido para ser o BASE do programa e que, em conjunto com a instrução USING, permite o endereçamento das instruções.

Linkage Conventions

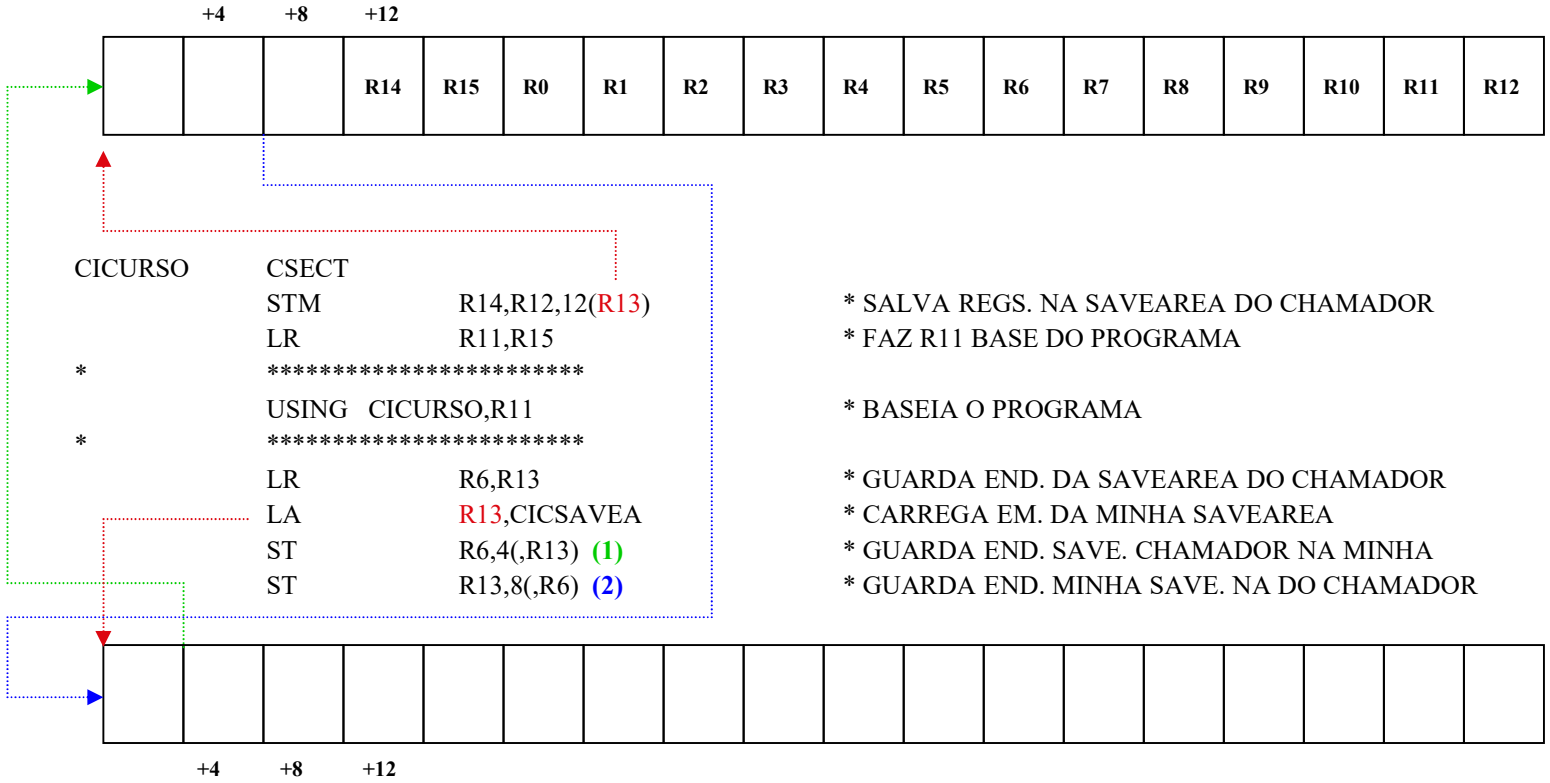
SAVEAREA

- Área de 72 bytes passada pelo programa chamador no R13
- Usada pelo programa chamado para salvar os registradores do chamador na entrada e restaurá-los na saída
- O programa chamado deve ter a sua SAVEAREA e estabelecer a ligação com a SAVEAREA anterior
- Ao finalizar, o programa chamado retorna ao programa chamador via R14

Início e término de um programa

Inicialização

- SAVEAREA programa chamador

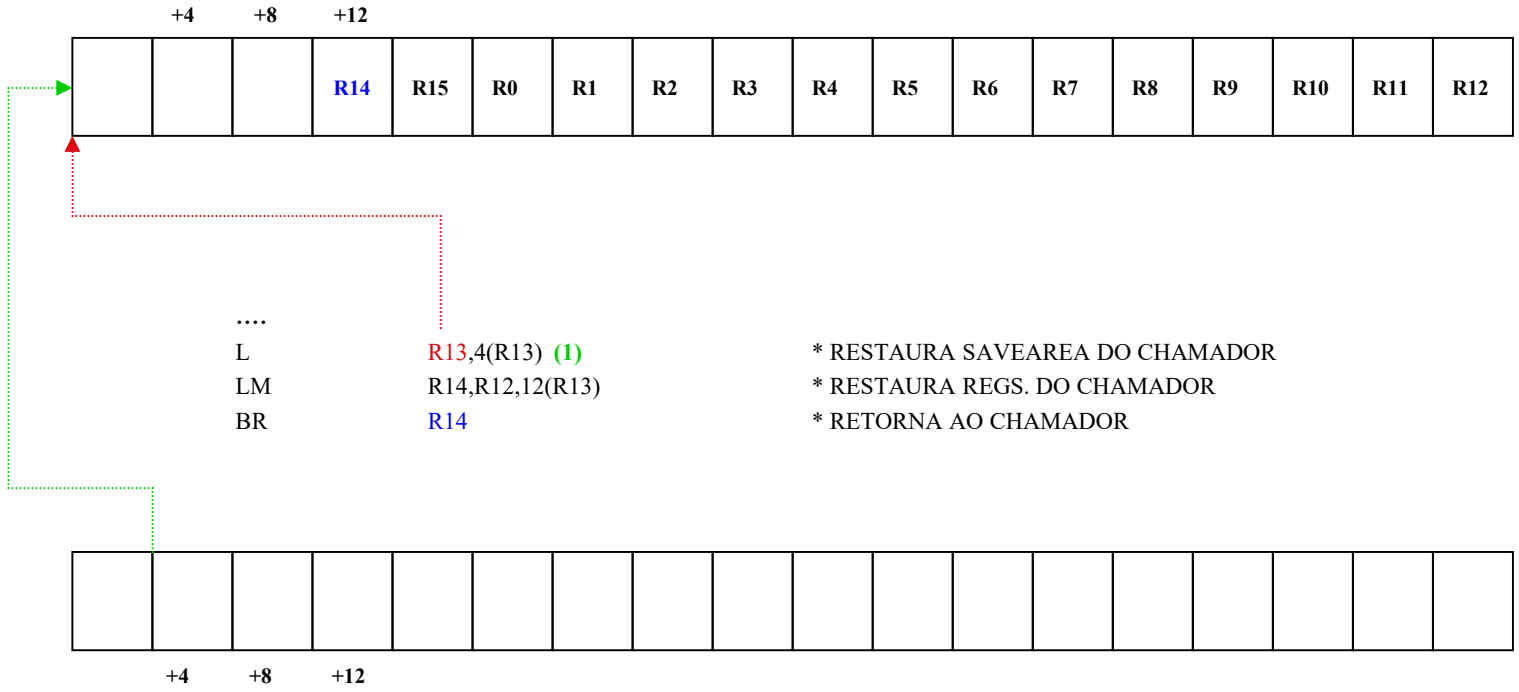


- SAVEAREA do meu programa (CICSAVEA)

Início e término de um programa

Finalização

- SAVEAREA programa chamador



- SAVEAREA do meu programa (CICSAVEA)

Instruções

- As instruções de máquina podem operar com **32** ou **64** bits.
- Neste curso trataremos somente de instruções com operação de 32 bits.
- Na apresentação da sintaxe das instruções, atentar para a numeração dos operandos. Por exemplo, na instrução: **ICM R1,M3,D2(B2)**, **M3** é tratado como terceiro parâmetro e não o segundo.
- Nas instruções gerais e nas aritméticas, os operandos são tratados como números inteiros com sinal.

Instruções

Carga de Registradores

- **LOAD ADDRESS (LA)**

  Carrega no registrador do primeiro operando o endereço do segundo operando (base + deslocamento + indexador).

  Tamanho - 4 bytes

  Formato - RX

  Sintaxe - LA R1,D2(X2,B2)

  Exemplo1 - Carregar no registrador 2 o endereço do campo REGENT:

  LA 2,REGENT

  Exemplo2 - Carregar o endereço 0 no registrador 15

  LA 15,0 ou **LA 15,0(0,0)**

Instruções

Carga de Registradores

- **LOAD (L)**

  Carrega no registrador do primeiro operando, 4 bytes obtidos a partir do endereço do segundo operando (base + deslocamento + indexador).

  Tamanho - 4 bytes

  Formato - RX

  Sintaxe - L R1,D2(X2,B2)

  Exemplo - Carregar no registrador 5 o endereço do campo AGE00100:

  L 5,AGE00100

  Supondo que AGE00100 esteja definido como :

  AGE00100 DC F'100'

  Após a execução da instrução o conteúdo do registrador 15 seria **00000064**

Instruções

Carga de Registradores

- **LOAD REGISTER(LR)**

  Carrega no registrador do primeiro operando o conteúdo do segundo operando.

  Tamanho - 2 bytes

  Formato - RR

  Sintaxe - LR R1,R2

  Exemplo - Carregar no registrador 8 o conteúdo do registrador 10:

  LR 8,10

Instruções

Carga de Registradores

- **LOAD HALFWORD(LH)**

  Carrega no registrador do primeiro operando dois bytes obtidos a partir do endereço do segundo operando (base + deslocamento + indexador). A carga é feita nos bits 16 a 31 do registrador e o bit 16 será propagado para os bits 0 a 15.

  Tamanho - 4 bytes
  Formato - RX
  Sintaxe - LH R1,D2(X2,B2)
  Exemplo - Carregar no registrador 12 o conteúdo do campo AGE0F0:
  LH 12,AGE0F0
  Supondo que **AGE0F0** esteja definido como:
  AGE0F0 DC X'F000'
  Após a execução da instrução, o conteúdo do registrador 12 seria:
  FFFFF000

Instruções

Carga de Registradores

- **LOAD MULTIPLE(LM)**

  Os registradores a partir de R1 até R3 são carregados com o conteúdo obtido no endereço de memória do segundo operando. Os registradores são carregados com sucessivos conjuntos de 4 bytes.

  Tamanho - 4 bytes

  Formato - RS

  Sintaxe - LM R1,R3,D2(X2,B2)

  Exemplo - Carregar nos registradores 2 a 5, o conteúdo do campo SAVEREGS:

  LM 2,5,SAVEREGS

  Supondo que o conteúdo do campo **SAVEREGS** seja:

  X'00000010008645340000000 10B2A3C5F'

  Após a execução da instrução, o conteúdo dos registradores seria:

  R2=00000010 R3=00864534 R4=00000001 R5=0B2A3C5F

Instruções

Descarga de Registradores

- **STORE (ST)**

  Guarda o conteúdo do registrador do primeiro operando no endereço dado pelo segundo operando (base + deslocamento + indexador).

  Tamanho - 4 bytes

  Formato - RX

  Sintaxe - ST R1,D2(X2,B2)

  Exemplo - Carregar o conteúdo do registrador 4 no campo SAVER4:

  ST 4,SAVER4

Instruções

Descarga de Registradores

- **STORE HALFWORD (STH)**

  Guarda os bits 16 a 31 do registrador do primeiro operando no endereço dado pelo segundo operando.

  Tamanho - 4 bytes

  Formato - RX

  Sintaxe - STH R1,D2(X2,B2)

  Exemplo - Guardar o conteúdo do registrador 8 (halfword) no campo SAVER8:

  STH 8,SAVER8

  Supondo que o conteúdo do registrador 8 seja X'80007FFF', após a execução da instrução, o conteúdo do campo SAVER8 será X'7FFF'.

Instruções

Aritmética

- **ADD FULLWORD (A)**

  Somar os 4 bytes obtidos no endereço do segundo operando ao conteúdo do registrador do primeiro operando. O resultado fica em R1.

  Tamanho - 4 bytes

  Formato - RX

  Sintaxe - A R1,D2(X2,B2)

  Exemplo - Somar o conteúdo do campo ACUMULA ao registrador 4:

  A 4,ACUMULA

  Supondo que o registrador 8 seja X'00007FFF' e ACUMULA seja X'00008000', após a execução da instrução, o conteúdo de R4 seria X'0000FFFF'. ACUMULA fica inalterado.

Instruções

Aritmética

- **ADD HALFWORD (AH)**

  Somar o 2 bytes obtidos no endereço do segundo operando ao conteúdo do registrador do primeiro operando. O resultado fica em R1. Antes da soma, o segundo operando é expandido para FULLWORD com a propagação do primeiro bit.

  Tamanho - 4 bytes

  Formato - RX

  Sintaxe - AH R1,D2(X2,B2)

  Exemplo - Somar o conteúdo do campo ACUMHALF ao registrador 5:

  AH 5,ACUMHALF

  Supondo que o registrador 5 seja X'000000FF' e ACUMHALF seja X'8000', após a execução da instrução, o conteúdo de R5 seria X'FFFF80FF'. ACUMHALF fica inalterado.

Instruções

Aritmética

- **SUBTRACT FULLWORD (S)**

  O conteúdo do segundo operando é subtraído do primeiro operando e o resultado é colocado no registrador do primeiro operando.

  Tamanho - 4 bytes

  Formato - RX

  Sintaxe - S R1,D2(X2,R2)

  Exemplo - Subtrair o conteúdo do campo ACUMULA do registrador 10:

  S 10,ACUMULA

  Supondo **R10=0000A000** e o campo **ACUMULA=X'00000880'**, após a execução da instrução: **R10=00009780**

Instruções

Aritmética

- **SUBTRACT HALFWORD (SH)**

  Subtrai 2 bytes obtidos no endereço do segundo operando do conteúdo do registrador do primeiro operando. O resultado fica em R1.

  Antes da subtração o operando 2 é expandido para FULLWORD com a propagação do primeiro bit.

  Tamanho - 4 bytes

  Formato - RX

  Sintaxe - SH R1,D2(X2,R2)

  Exemplo - Subtrair o conteúdo do campo ACUMHALF do registrador 5:

  SH 5,ACUMHALF

  Supondo **R5=00010000** e o campo **ACUMHALF=X'8002'**, após a execução da instrução: **R5=00017FFE e ACUMHALF** fica inalterado.

Instruções

Aritmética

- **SUBTRACT REGISTER (SR)**

  O conteúdo do registrador do segundo operando é subtraído do registrador do primeiro operando e o resultado fica no primeiro operando.

  Tamanho - 2 bytes

  Formato - RR

  Sintaxe - SR R1, R2

  Exemplo - Subtrair o conteúdo do registrador R4 do registrador R5 e colocar o resultado em R5:

  SR R5,R4

Instruções

Aritmética

- **MULTIPLY HALFWORD (MH)**

  O multiplicando deve estar num registrador geral qualquer no primeiro operando.

  O multiplicador são 2 bytes obtidos no endereço de memória dado pelo operando 2.

  O resultado fica no registrador geral R1.

  Tamanho - 4 bytes

  Formato - RX

  Sintaxe - MH R1,D2(X2,B2)

  Exemplo - Supondo que R2=00002000 e CONT=x'0100' após a instrução:

  MH 2,CONT

  Teremos: R2=00200000 e 'CONT' permanece inalterado.

Instruções

Aritmética

- **MULTIPLY FULLWORD (M)**

  O multiplicando deve estar num registrador ímpar de um conjunto par-impar de registradores (R1 e R1+1), onde R1 é o registrador par.

  O multiplicador são 4 bytes obtidos no endereço de memória dado pelo operando 2.

  O resultado fica no conjunto de registradores par-ímpar como um único número de 64 bits.

  Tamanho - 4 bytes

  Formato - RX

  Sintaxe - M R1,D2(X2,B2)

  Exemplo - Supondo que R2=00000000, R3=00002000 e CONT=x'00000100' após a instrução:

  M 2,CONT

  Teremos: R2=00000000, R3=00200000 e 'CONT' permanece inalterado.

Instruções

Aritmética

- **MULTIPLY REGISTER (MR)**

  O multiplicando deve estar num registrador ímpar de um conjunto par-impar de registradores (R1 e R1+1), onde R1 é o registrador par.

  O operando 2 é o multiplicador e deve ser um registrador geral qualquer.

  O resultado fica no conjunto de registradores par-ímpar do operando 1.

  Tamanho - 2 bytes

  Formato - RR

  Sintaxe - MR R1,R2

  Exemplo - Supondo que R4=00000000, R5=00000110 e R6=000000A0 após a instrução:

  MR 4,6

  Teremos: R4=00000000, R5=0000AA00 e R6 permanece inalterado.

Instruções

Aritmética

- **DIVIDE (D)**

  O dividendo é o operando 1 que é considerado um número de 64 bits armazenado num conjunto par-impar de registradores (R1 e R1+1), onde R1 é o registrador par. O número é ajustado à direita no conjunto par-ímpar.

  O divisor são os 4 bytes obtidos pelo endereço de memória dado pelo operando 2.

  O quociente fica no registrador ímpar do conjunto e o resto da divisão no registrador par.

  Tamanho - 4 bytes
  Formato - RX
  Sintaxe - D R1,D2(X2,B2)
  Exemplo - Supondo que R6=00000000, R7=0000A001 e o campo TOTAL=X'00000100'. Após a instrução:

  D 6,TOTAL

  Teremos: R6=00000001, R7=000000A0 e TOTAL permanece inalterado.

Instruções

Aritmética

- **DIVIDE REGISTER (DR)**

  O dividendo é o operando 1 que é considerado um número de 64 bits armazenado num conjunto par-impar de registradores (R1 e R1+1), onde R1 é o registrador par. O número é ajustado à direita no conjunto par-ímpar.

  O divisor é um registrador geral informado no operando 2.

  O quociente fica no registrador ímpar do conjunto e o resto da divisão no registrador par.

  Tamanho - 2 bytes

  Formato - RR

  Sintaxe - DR R1,R2

  Exemplo - Supondo que R8=00000000, R9=0000A00A e R10=00000100. Após a instrução:

  DR 8,10

  Teremos: R8=0000000A, R9=000000A0 e R10 permanece inalterado.

Instruções

Código de condição

- Posicionado pelas instruções de máquina (bits 18 e 19 da PSW)
- Valores que pode assumir:

Binário	Decimal
00	0
01	1
10	2
11	3

- Para cada CC existe uma máscara que pode ser utilizada nas instruções de desvio

  CC=0 => Máscara = 8
  CC=1 => Máscara = 4
  CC=2 => Máscara = 2
  CC=3 => Máscara = 1

- Exemplo: Para todas as instruções de soma e subtração

  Resultado = 0 => CC=0
  Resultado < 0 => CC=1
  Resultado > 0 => CC=2
  Estouro => CC=3

Instruções

Código de condição

- Exemplo de aplicação prática: Após a soma dos registradores 5 e 6, desejo desviar para a rotina ROTERRO se o resultado for negativo.

  As instruções para isso seriam:

  AR 5,6
  BC 4,ROTERRO

  Se eu desejasse o desvio para um resultado menor ou igual a zero:

  AR 5,6
  BC 12,ROTERRO

  Existem mneumônicos para substituir as máscaras:

BZ	=> Desvia se o resultado for 0.	Equivale ao BC 8
BP	=> Desvia se o resultado for positivo.	Equivale ao BC 2
BM	=> Desvia se o resultado for negativo.	Equivale ao BC 4
BO	=> Desvia se houver overflow.	Equivale ao BC 1
BNP	=> Desvia se o resultado não for positivo.	Equivale ao BC 13

Instruções

Comparação e Desvio

- **BRANCH ON CONDITION (BC)**

  O operando 1 é a máscara e o operando 2 o endereço de desvio.

  A máscara é setada em função do desvio pretendido, de acordo com o Condition Code (CC).

  Tamanho: 4 bytes

  Formato: RX

  Sintaxe: **BC M1,D2(X2,B2)**

- **BRANCH ON CONDITION REGISTER (BCR)**

  O operando 1 é a máscara e o operando 2 um registrador contendo o endereço de desvio.

  Tamanho: 4 bytes

  Formato: RX

  Sintaxe: **BCR M1,R2**

Instruções

Comparação e Desvio

- **BRANCH AND LINK (BAL) e BRANCH AND SAVE (BAS)**

  O endereço da próxima instrução é guardado no registrador do primeiro operando e é feito um desvio incondicional para o endereço do segundo operando.

  Tamanho: 4 bytes

  Formato: RX

  Sintaxe: **BAL R1,D2(X2,B2) ou BAS R1,D2(X2,B2)**

- **BRANCH AND LINK REGISTER (BALR) e BRANCH AND SAVE REGISTER (BASR)**

  O endereço da próxima instrução é guardado no registrador do primeiro operando e é feito um desvio incondicional para o endereço apontado pelo registrador do segundo operando.

  Tamanho: 2 bytes

  Formato: RR

  Sintaxe: **BALR R1,R2 ou BASR R1,R2**

Instruções

Comparação e Desvio

- **MNEUMÔNICOS**

  Mneumônicos podem ser utilizados para substituir a máscara após as instruções que setam o código de condição (CC).

  Desvio incondicional: **B** ou **BR**

  Após instruções de comparação:

BH ou BHR	- desvia quando o primeiro operando é maior
BL ou BLR	- desvia quando o primeiro operando é menor
BE ou BER	- desvia quando os operandos são iguais
BNH ou BNHR	- desvia quando o primeiro operando não é maior
BNL ou BNLR	- desvia quando o primeiro operando não é menor
BNE ou BNER	- desvia quando os operandos são diferentes

Instruções

Comparação e Desvio

- **MNEUMÔNICOS**

  Após instruções aritméticas:

  BP ou BPR - desvia quando o resultado é positivo
  BM ou BMR - desvia quando o resultado é negativo
  BNP ou BNPR - desvia quando o resultado não é positivo
  BNM ou BNMR - desvia quando o resultado não é negativo
  BNZ ou BNZR - desvia quando o resultado não for zero
  BZ ou BZR - desvia quando o resultado for zero

  Após instruções de teste:

  BO ou BOR - desvia se todos os bits da máscara estiverem ligados
  BM ou BMR - desvia se há bits da máscara ligados e desligados
  BNO ou BNOR - desvia se nenhum bit da máscara estiver ligado
  BZ ou BZR - desvia quando todos os bits da máscara estiverem desligados

Instruções

Lógicas

- Similares às instruções aritméticas
- O posicionamento do código de condição (CC) é feito da seguinte forma:

  CC = 0 => resultado = 0 e não “vai um”
  CC = 1 => resultado # 0 e não “vai um”
  CC = 2 => resultado = 0 e “vai um”
  CC = 3 => resultado # 0 e “vai um”

- As instruções lógicas não acusam estouro (overflow)
- As operações lógicas não levam em conta o bit de sinal

Instruções

Lógicas

- **ADD LOGICAL FULLWORD (AL)**

  Soma os 4 bytes obtidos no endereço do segundo operando ao conteúdo do registrador do primeiro operando. O resultado fica em R1.

  Tamanho: 4 bytes

  Formato: RX

  Sintaxe: AL R1,D2(X2,B2)

  Exemplo: Somar o conteúdo do campo ACUMULA ao registrador 6

  AL 6,ACUMULA

  Supondo que R6=7FFFFFFF e ACUMULA=X'80000001'

  Após a execução da instrução: R6=00000000, o campo ACUMULA fica inalterado e o CC = 2.

Instruções

Lógicas

- **ADD LOGICAL REGISTER (ALR)**

  Os conteúdos dos registradores são somados e o resultado fica no registrador do primeiro operando.

  Tamanho: 2 bytes

  Formato: RR

  Sintaxe: ALR R1,R2

  Exemplo: Somar o conteúdo dos registradores 8 e 9 e colocar o resultado no registrador 9

  ALR 9,8

Instruções

Lógicas

- **SUBTRACT LOGICAL FULLWORD (SL)**

  O conteúdo dos 4 bytes endereçados pelo segundo operando é subtraído do registrador do primeiro operando e o resultado fica em R1.

  Tamanho: 4 bytes

  Formato: RX

  Sintaxe: SL R1,D2(X2,B2)

  Exemplo: Subtrair o conteúdo do campo CONTA2 do registrador 7 e colocar o resultado no registrador 7

  SL 7,CONTA2

Instruções

Lógicas

- **SUBTRACT LOGICAL REGISTER (SLR)**

  O conteúdo do registrador do segundo operando é subtraído do registrador do primeiro operando.

  Tamanho: 2 bytes

  Formato: RR

  Sintaxe: SLR R1,R2

  Exemplo: Subtrair o conteúdo do registrador 7 do registrador 3 e colocar o resultado no registrador 3

  SLR 3,7

Instruções

Lógicas

- **Instruções de comparação**

  Os dados a serem comparados são considerados binários sem sinal

  A comparação é feita bit a bit da esquerda para a direita

  O posicionamento do código de condição (CC) é feito da seguinte maneira:

  CC=0 => Operando 1 = operando 2

  CC=1 => Operando 1 < operando 2

  CC=2 => Operando 1 > operando 2

  CC=3 => não utilizado

Instruções

Lógicas

- **COMPARE LOGICAL FULLWORD (CL)**

  O primeiro operando é um registrador geral que será comparado com o conteúdo obtido no endereço fornecido pelo segundo operando.

  Tamanho: 4 bytes

  Formato: RX

  Sintaxe: CL R1,D2(X2,B2)

  Exemplo: Comparar o registrador 9 com o conteúdo do campo LIMITE e desviar para o endereço FIMLOOP se os dois forem iguais.

```
CL   9,LIMITE
BE   FIMLOOP
```

Instruções

Lógicas

- **COMPARE LOGICAL REGISTER (CLR)**

  O primeiro operando é um registrador geral que será comparado com o registrador geral do segundo operando.

  Tamanho: 2 bytes

  Formato: RR

  Sintaxe: CLR R1,R2

  Exemplo: Comparar os registradores 10 e 11 e desviar para o endereço FIM se o conteúdo de R10 for maior que o de R11.

  CLR 10,11

  BH FIM

Instruções

Lógicas

- **COMPARE LOGICAL CHARACTER (CLC)**

  Os dois operandos são endereços de memória que serão comparados byte a byte.

  O tamanho a ser comparado é o valor de L (bits 8 a 15) da instrução. O tamanho pode ser implícito na instrução (tamanho do primeiro operando).

  Tamanho: 6 bytes

  Formato: SS

  Sintaxe: CLC D1(L,B1),D2(B2)

  Exemplo: Comparar os campos CAMPO1 e CAMPO2 e desviar para o endereço DIFERE se os conteúdos não forem iguais.

```
CLC   CAMPO1,CAMPO2
BNE   DIFERE
```

Instruções

Lógicas

- **COMPARE LOGICAL IMADIATE (CLI)**

  O byte obtido no endereço dado pelo operando 1 é comparado ao operando 2. O operando 2 é imediato, ou seja, faz parte da instrução.

  Tamanho: 4 bytes

  Formato: SI

  Sintaxe: CLI D1(B1),I2

  Exemplo: Comparar o campo AGE0002 com o caracter ‘2’. Se forem diferentes desviar para o endereço FINALIZAR.

```
CLI   AGE0002,C'2'
BNE   FINALIZA
```

Instruções

Lógicas

- **COMPARE LOGICAL CHARACTER UNDER MASK (CLM)**

  O operando M3 é uma máscara de 4 bits. A máscara indica, da esquerda para a direita, de um a quatro bytes do registrador do operando 1 que deve(m) ser comparado(s) com o mesmo número de bytes dados pelo endereço do operando 2.

  Tamanho: 4 bytes

  Formato: RS

  Sintaxe: CLM R1,M3,D2(B2)

  O código de condição (CC) é posicionado da seguinte forma:

  CC=0 => bytes selecionados em R1 são iguais aos do operando 2

  CC=1 => bytes selecionados em R1 < operando 2

  CC=2 => bytes selecionados em R1 > operando 2

  Exemplo: Comparar o segundo e o terceiro bytes do registrador 8 com o conteúdo do campo TESTREG.

  CLM 8,6,TESTREG ou CLM 8,B'0110',TESTREG

Instruções

Lógicas

- **INSERT CHARACTER (IC)**

  Insere um byte endereçado pelo segundo operando nos bits 24 a 31 do registrador indicado no primeiro operando. Os demais bits do registrador ficam inalterados.

  Tamanho: 4 bytes

  Formato: RX

  Sintaxe: IC R1,D2(X2,B2)

  Exemplo: Supondo o conteúdo de R3=00112200 e a constante TRESTRES DC X'33'

  IC 3,TRESTRES

  Após a execução da instrução acima: R3=00112233

Instruções

Lógicas

- **INSERT CHARACTER UNDER MASK (ICM)**

  O operando M3 é uma máscara de 4 bits. A máscara indica, da esquerda para a direita, de um a quatro byte(s) que deve(m) ser inserido(s) no registrador do operando 1 a partir do endereço dado pelo operando 2.

  Tamanho: 4 bytes

  Formato: RS

  Sintaxe: ICM R1,M3,D2(B2)

  O código de condição é posicionado da seguinte forma:

  CC=0 => todos os bits carregados são zero ou a máscara é zero

  CC=1 => o primeiro bit carregado é um (número negativo)

  CC=2 => o primeiro bit carregado é zero e existe, pelo menos, mais um bit 1 (número é positivo)

  Exemplo: Carregar o primeiro byte do registrador 6 com o conteúdo x'00'.

  Supondo: ZEROS DC X'00'

  ICM 6,8,ZEROS ou ICM 6,B'1000',ZEROS

Instruções

Lógicas

- **STORE CHARACTER (STC)**

  Insere os bits 24 à 31 do registrador indicado no operando 1 no endereço indicado pelo segundo operando.

  Tamanho: 4 bytes

  Formato: RX

  Sintaxe: STC R1,D2(X2,B2)

  Exemplo: Supondo o conteúdo de R5=111111FF a a variável:

  VARIA1 DS X

  STC R5,VARIA1

  Após a execução da instrução, VARIA1=X'FF'

Instruções

Lógicas

- **STORE CHARACTER UNDER MASK (STCM)**

  O operando M3 é a máscara de 4 bits. A máscara indica, da esquerda para a direita, de um a quatro byte(s) que deve(m) ser inserido(s) no endereço dado pelo operando 2 a partir do registrador do primeiro operando.

  Tamanho: 4 bytes
  Formato: RS
  Sintaxe: STCM R1,M3,D2(B2)
  Exemplo: Supondo o conteúdo de R5=112211FF a a variável:

  VARIA2 DS XL2

  STCM R5,12,VARIA2 ou STCM R5,B'1100',VARIA2

  Após a execução da instrução, VARIA2=X'1122'

Instruções

Lógicas

- **MOVE IMMEDIATE (MVI)**

  O byte informado no operando imediato I2 é movido para o endereço dado pelo operando 1.

  Tamanho: 4 bytes

  Formato: SI

  Sintaxe: MVI D1(B1),I2

  Exemplo: Mover o caracter "@" para o campo FIMREG

  MVI FIMREG,C'@'

Instruções

Lógicas

- **MOVE CHARACTER (MVC)**

  Os dados endereçados pelo operando 2 são movidos para o endereço apontado pelo operando 1. O número de bytes a ser movido é indicado em 'L' que pode variar de 1 a 256 e pode ser implícito ou explícito.

  A movimentação é feita byte a byte, da esquerda para a direita.

  Tamanho: 6 bytes
  Formato: SS
  Sintaxe: MVC D1(L,B1),D2(B2)
  Exemplo: Supondo:

  CAMPO1 DC CL5'12345'
  CAMPO2 DC CL8'ABCDEFGH'

  Após a instrução: **MVC CAMPO2(5),CAMPO1**
  O conteúdo do CAMPO2 passa a ser '12345FGH'

Instruções

Lógicas

- **ÁLGEBRA BOOLEANA - EXEMPLOS**

1) Limpar o registrador R2

Supondo: R2=A2B1C4DF

XR R2,R2

1010	0010	1011	0001	1100	0100	1101	1111
1010	0010	1011	0001	1100	0100	1101	1111
0000	0000	0000	0000	0000	0000	0000	0000

2) Ligar o primeiro bit do registrador R6

Supondo: R6=0025465F

MASK2 DC X'80000000'

O 6,MASK2

R6=8025465F

0000	0000	0010	0101	0100	0110	0101	1111
1000	0000	0000	0000	0000	0000	0000	0000
1000	0000	0010	0101	0100	0110	0101	1111

Instruções

Lógicas

- **TEST UNDER MASK (TM)**

  Testa os bits do byte endereçado pelo primeiro operando de acordo com os bits ligados da máscara do segundo operando. Os bits testados são os ligados na máscara.

  O código de condição é setado da seguinte forma:

  CC=0 => todos os bits testados são zero
  CC=1 => bits testados são zero e um
  CC=2 => todos os bits testados são um

  Tamanho: 4 bytes
  Formato: SI
  Sintaxe: TM D1(B1),M2
  Exemplo: Se o primeiro bit do campo BCT101 estiver ligado, desviar para ERROSTA:

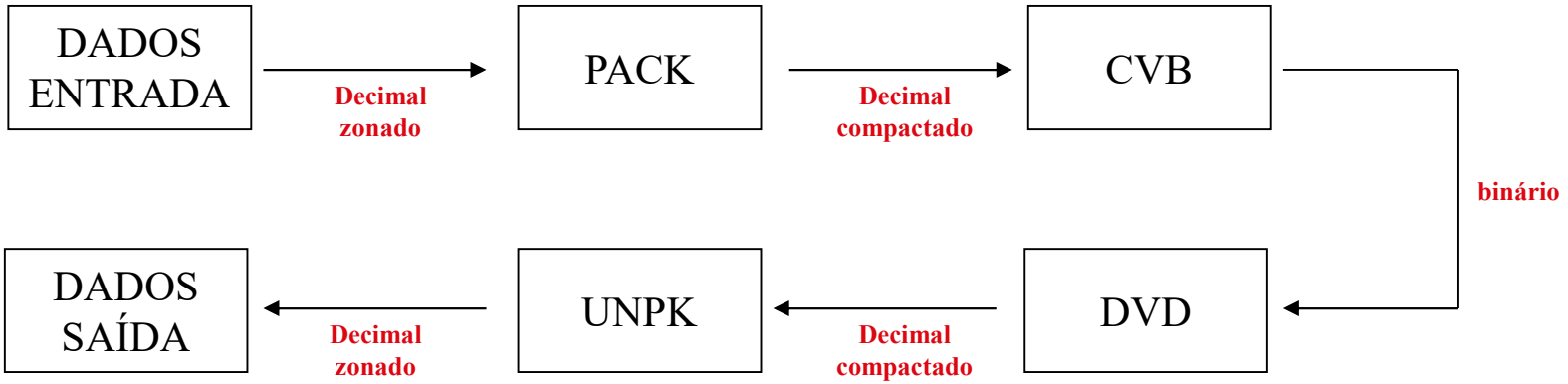
  TM BCT101,X'80'
  BO ERROSTA (ou BC 1,ERROSTA)

Instruções

Decimais

- **CONVERSÃO DE FORMATOS**

PACK => converte de decimal zonado para decimal compactado
UNPK => converte de decimal compactado para decimal zonado
CVB => converte de decimal compactado para binário
CVD => converte de binário para decimal compactado



Instruções

Decimais

- **PACK**

  Converte os dados em formato zonado do endereço apontado pelo operando 2 para decimal compactado no endereço apontado pelo operando 1.

  Os dois operandos podem ter tamanho implícito ou explícito (L1 e L2).

  A conversão é feita da direita para a esquerda. Pode haver truncamento ou preenchimento com zeros, dependendo dos tamanhos de cada campo.

  Tamanho: 6 bytes
  Formato: SS
  Sintaxe: PACK D1(L1,B1),D2(L2,B2)
  Exemplo: **PACK DESTCP,ORIGEMDZ**
  ORIGEMDZ DC CL5'34724' => **F3 F4 F7 F2 F4**

  DESTCP DC CL4 **00 34 72 4C** **(sinal)**

  (preenchimento)

Instruções

Decimais

- **UNPACK (UNPK)**

  Converte os dados em formato decimal compactado do endereço apontado pelo operando 2 para decimal zonado no endereço apontado pelo operando 1.

  Os dois operandos podem ter tamanho implícito ou explícito (L1 e L2).

  A conversão é feita da direita para a esquerda. Pode haver truncamento ou preenchimento com zeros, dependendo dos tamanhos de cada campo.

  Tamanho: 6 bytes

  Formato: SS

  Sintaxe: UNPK D1(L1,B1),D2(L2,B2)

  Exemplo: **UNPK DESTCZ,ORIGEMDP**

  ORIGEMDP DC PL5'23542' **=> 00 00 23 54 2C → (sinal)**

  DESTCZ DC CL5 **F2 F3 F5 F4 F2**

Instruções

Decimais

- **CONVERT TO DECIMAL (CVD)**

  O número em binário contido no registrador R1 é convertido para o formato decimal compactado no endereço de memória apontado pelo operando 2.

  O número convertido ocupa 8 bytes da memória.

  Tamanho: 4 bytes
  Formato: RX
  Sintaxe: CVD R1,D2(X2,B2)
  Exemplo: **CVD R4,DOUBLEW**

  Supondo: R4=00000100

  Após a execução da instrução teremos: **DOUBLEW = X'000000000000256C'**

Instruções

Decimais

- **CONVERT TO BINARY (CVB)**

  O número em decimal compactado contido no endereço dado pelo operando 2 é convertido para o formato binário no registrador do primeiro operando.

  O número convertido ocupa 8 bytes da memória.

  Tamanho: 4 bytes
  Formato: RX
  Sintaxe: CVB R1,D2(X2,B2)
  Exemplo: Supondo o campo DOUBLEP = X'000000000004096C'
  CVB R5,DOUBLEP

  Após a execução da instrução teremos: **R5= X'00001000'**

Instruções

Decimais

- **ADD PACKED (AP)**

  O conteúdo do campo endereçado pelo operando 2 é somado ao conteúdo do campo endereçado pelo operando 1 e o resultado é colocado no operando 1. Os dois campos têm que estar no formato decimal compactado.

  O tamanho de ambos os campos pode ser implícito ou explícito. Se o tamanho do primeiro operando não for suficiente, será setado CC=3.

  Tamanho: 6 bytes
  Formato: SS
  Sintaxe: AP D1(L1,B1),D2(L2,B2)
  Exemplo:

  ```
  CONTA   DC      PL3'6'          => X'00006C'
  CONTB   DC      PL4'1234'       => X'0001234C'
  ```

  Após a instrução:

  ```
  AP      CONTB,CONTA
  ```

  CONTB = X'0001240C'

Instruções

Decimais

- **SUBTRACT PACKED (SP)**

  O conteúdo do campo endereçado pelo operando 2 é subtraído do conteúdo do campo endereçado pelo operando 1 e o resultado é colocado no operando 1. Os dois campos têm que estar no formato decimal compactado.

  O tamanho de ambos os campos pode ser implícito ou explícito.

  Tamanho: 6 bytes
  Formato: SS
  Sintaxe: SP D1(L1,B1),D2(L2,B2)
  Exemplo:

  PARCIALDC PL3'50' => X'00050C'
  TOTAL DC PL4'150' => X'0000150C'

  Após a instrução:

  SP TOTAL,PARCIAL

  TOTAL = X'0000100C'

Instruções

Decimais

- **ZERO AND ADD PACKED (ZAP)**

  O conteúdo do campo endereçado pelo operando 1 é zerado e, em seguida, o conteúdo do campo endereçado pelo operando 2 é somado ao conteúdo do campo endereçado pelo operando 1. Os dois campos têm que estar no formato decimal compactado.

  O tamanho de ambos os campos pode ser implícito ou explícito.

  Tamanho: 6 bytes
  Formato: SS
  Sintaxe: ZAP D1(L1,B1),D2(L2,B2)
  Exemplo:

  PARCIALDC PL3'0' => X'00000C'
  TOTALP DC PL4'150' => X'0000150C'

  Após a instrução:

  ZAP TOTALP,PARCIAL

  TOTALP = X'0000000C'

Instruções

Decimais

- **COMPARE PACKED (ZAP)**

  O conteúdo dos campos endereçados pelos operandos 1 e 2 são comparados algebricamente. O código de condição é setado da seguinte forma:

  CC=0 => operandos são iguais
  CC=1 => operando 1 < operando 2
  CC=2 => operando 1 > operando 2

  Tamanho: 6 bytes
  Formato: SS
  Sintaxe: CP D1(L1,B1),D2(L2,B2)
  Exemplo:

  PACK1 DC PL3'10' => X'00010C'
  PACK2 DC PL4'15' => X'0000015C'

  Após a instrução:

  CP PACK1,PACK2

  CC = 1

Instruções

Edição

- **EDIT (ED)**

  O operando 1 aponta para uma área de memória onde está o modelo sobre o qual a edição vai funcionar. O operando 2 aponta uma área de memória que deve estar em formato decimal compactado, onde está o campo fonte da edição.

  O conteúdo do campo fonte é convertido para formato zonado sob o controle do campo modelo e o resultado fica na área onde estava o modelo. Os bytes são processados um a um da esquerda para a direita.

  O modelo possui caracteres de controle para a edição. O primeiro byte é o caracter de preenchimento.

  O indicador de significância (x'21') é uma chave que indica se os caracteres do campo fonte subsequentes são significativos. Isto é levado em conta para decidir o resultado do dígito a ser editado.

  O separador de campos (x'22') identifica campos individuais, como por exemplo, uma vírgula separando centavos. É substituído pelo caracter de preenchimento e desligado o indicador de significância.

Instruções

Edição

- **EDIT (ED)**

  O seletor de dígitos (x'20') indica que o dígito do campo fonte deve ser convertido a zonado no campo modelo, a não ser que o indicador de significância esteja desligado e o dígito seja zero. Nesse caso será substituído pelo caracter de preenchimento.

  O número de bytes resultante é igual ao tamanho implícito ou explícito do primeiro operando (modelo).

  Tamanho: 6 bytes

  Formato: SS

  Sintaxe: ED D1(L1,B1),D2(B2)

  Exemplo: Transformar o conteúdo do campo VALORPK para o formato xxx.xxx,xx sem ZEROS à esquerda no campo SAIDAED.

  ```
  VALORPK     DC     PL4'125000'
  SAIDAED     DC     CL10'4020204B2020216B2020'
  ```

  Após a instrução:

  ```
  ED      SAIDAED,VALORPK
  ```

  SAIDAED = '1.250,00' ou X'4040F14BF2F5F06BF0F0'

Instruções

Edição – Tabela prática

Byte do modelo	Status Indicador Significância	Dígito na fonte	4 bits da fonte positivo	Byte resultante	Status Indicador Significância
X'20' (seletor de dígito)	OFF	0	(1)	Preenchimento	OFF
		1 – 9	Não	Dígito da fonte	ON
		1 – 9	Sim	Dígito da fonte	OFF
	ON	0 – 9	Não	Dígito da fonte	ON
		0 – 9	Sim	Dígito da fonte	OFF
X'21' (indicador significância)	OFF	0	Não	Preenchimento	ON
		0	Sim	Preenchimento	OFF
		1 – 9	Não	Dígito da fonte	ON
		1 – 9	Sim	Dígito da fonte	OFF
	ON	0 – 9	Não	Dígito da fonte	ON
		0 – 9	Sim	Dígito da fonte	OFF
X'22' (separador de campos)	(1)	(2)	(2)	Preenchimento	OFF
Qualquer outro caracter	OFF	(2)	(2)	Preenchimento	OFF
	ON	(2)	(2)	Caracter	ON
(1) => Sem efeito no byte de significância			**(2) => dígito fonte não é considerado**		

Instruções

Desvio e controle de LOOP

- **BRANCH ON COUNT (BCT)**

  O primeiro operando é o registrador R1, utilizado como contador de repetições. A cada execução da instrução, é subtraído um do registrador R1 e desviado para o endereço indicado no operando 2.

  Quando o conteúdo do registrador R1 for igual a zero, não ocorre o desvio, e a instrução seguinte do programa é executada.

  Tamanho: 4 bytes

  Formato: RX

  Sintaxe: BCT R1,D2(X2,B2)

  Exemplo: Executar 10 vezes o trecho do programa iniciado em LOOP1.

```
          LA      8,10
LOOP1     EQU *
          ... (instruções que serão executadas 10 vezes)
          BCT     8,LOOP1
```

Macro Instruções

- **O que são MACRO INSTRUÇÕES**

  Conjunto de comandos e/ou instruções, codificados diretamente no programa ou em uma biblioteca específica (MACLIB), que são tratados em tempo de compilação e podem ou não gerar instruções a serem executadas durante o processamento.

  Existem inúmeros comandos usados na geração de macros que constituem uma linguagem de programação própria (AIF, AGO, SETA, SETC, LCLA, etc.).

  Seu uso mais comum, é quando se tem uma sequência padronizada de instruções que se repetem.

  Outra utilização importante, é para gerar instruções de máquina distintas dependendo de diferentes parâmetros recebidos.

  Exemplo:

Gera o seguinte código:

```
INICIO   SAVE    (14,12)
+        DS      0H
+        STM     14,12,12(13)
```


No programa compilado, indica código gerado por MACRO

Macro Instruções

- **Algumas MACROS muito utilizadas**

SAVE	- Salva registradores
RETURN	- Restaura registradores e retorna
GETMAIN	- Obtém área de memória virtual
FREEMAIN	- Libera memória virtual
STORAGE	- Obtém e libera memória virtual
WTO	- Emite mensagem ao Operador
SNAP	- Lista áreas de memória, registradores, etc.
ABEND	- Termina anormalmente uma TASK

Macro Instruções

DATA CONTROL BLOCK (DCB)

Define o bloco de controle usado pelo método de acesso para que um programa possa fazer operações de I/O em um arquivo.

Métodos de acesso que utilizam a DCB: QSAM, QISAM, BDAM, BPAM, BSAM E BISAM.

Sintaxe: Principais parâmetros para arquivo sequencial (QSAM)

```
DCB     DSORG=PS,MACRF=opção,
        DDNAME=ddname,EODAD=endrotina
```

Onde:

opção => G=GET, P=PUT, L=LOCATE, M=MOVE

ddname => nome do arquivo que deverá ser informado no JCL ou alocado dinamicamente via macro DYNALLOC

endrotina => endereço da rotina que ganhará o controle ao fim do arquivo

Exemplo:

```
SAIDA   DCB     DDNAME=SAIDA,DSORG=PS,MACRF=(PM),
                EODAD=FIM_ARQ
```

Macro Instruções

OPEN e CLOSE

- **OPEN**

  Prepara o arquivo para operações de I/O preenchendo alguns campos da DCB

  Sintaxe: Principais parâmetros para arquivo sequencial (QSAM)

  OPEN (dcbaddr,(opção))

  Onde:

  dcbaddr => endereço da DCB do arquivo

  opção => opções de processamento (INPUT,OUTPUT)

  endrotina => endereço da rotina que ganhará o controle ao fim do arquivo

  Exemplo:

  OPEN (SAIDA,(OUTPUT))

- **CLOSE**

  Sintaxe: **CLOSE (dcbaddr)**

  Onde:

  dcbaddr => endereço da DCB do arquivo

  Exemplo:

  CLOSE (SAIDA)

Macro Instruções

GET

Lê o registro de um arquivo. Para Para o tipo ‘LOCATE’ o registrador 1 aponta para o registro lido; para o tipo ‘MOVE’ devemos indicar a área onde será lido o registro.

Sintaxe: Para o tipo ‘LOCATE’

```
GET     dcbaddr
```

Onde:

dcbaddr => endereço da DCB do arquivo

Exemplo:

```
GET     (ARQENT)
```

Sintaxe: Para o tipo ‘MOVE’

```
GET     dcbadd,endaddr
```

Onde:

dcbaddr => endereço da DCB do arquivo

endaddr => endereço da área que receberá o registrao lido

Exemplo:

```
GET     ARQENT,BUFFERI
```

Macro Instruções

PUT

Grava um registro em um arquivo. Para o tipo ‘LOCATE’ o registrador 1 deve apontar para o registro a ser gravado; para o tipo ‘MOVE’ devemos indicar a área de onde será gravado o registro.

Sintaxe: Para o tipo ‘LOCATE’

```
PUT     dcbaddr
```

Onde:

dcbaddr => endereço da DCB do arquivo

Exemplo:

```
PUT     (ARQSAI)
```

Sintaxe: Para o tipo ‘MOVE’

```
PUT     dcbadd,endaddr
```

Onde:

dcbaddr => endereço da DCB do arquivo

endaddr => endereço da área de onde será gravado o registro

Exemplo:

```
PUT     ARQSAI,BUFFERI
```

Macro Instruções

CALL

Passa o controle a uma CONTROL SECTION (CSECT) no endereço especificado.

Ao término da CSECT chamada, o programa chamador espera ganhar o controle novamente na instrução seguinte ao CALL.

Se a CSECT chamada tiver em outro módulo, este é linkeditado junto com o módulo chamador.

Sintaxe: **CALL entryname**

Onde:

entryname => nome da CSECT para onde será desviado o controle

Exemplo:

```
CALL    CICP0000
```

Macro Instruções

LINK

Passa o controle, sincronamente, para uma entryname em um outro módulo de carga.

A entryname que receberá o controle deve ser um membro de uma biblioteca de módulos ou um alias em um diretório de um particionado.

Sintaxe: **LINK EP=entryname**

LINK EPLOC=entryname address

LINK DE=listname address

Onde:

entryname => nome do módulo que ganhará o controle

entryname address => endereço do módulo que ganhará o controle

listname address => endereço do módulo obtido através da chamada anterior da macro BLDL

Exemplo:

LINK EP=SBSTCK

Macro Instruções

LOAD

Carrega um módulo para a memória virtual. Após a carga, o registrador 0 conterá o endereço do módulo carregado.

O controle não é passado para o módulo carregado. A macro DELETE tira o módulo da memória.

Sintaxe: **LOAD EP=entryname**

LOAD EPLOC=entryname address

LOAD DE=listname address

Onde:

entryname => nome do módulo que ganhará o controle

entryname address => endereço do módulo que ganhará o controle

listname address => endereço do módulo obtido através da chamada anterior da macro BLDL

Exemplo:

LOAD EP=SIAD

Macro Instruções

ATTACH

Cria uma nova task (TCB) e passa o controle para a nova task no endereço apontado.

A nova task passa a ser uma subtask da task mãe e seu processamento é independente daquela, porém, se atask mãe abendar, ela também abenda.

Usada quando necessitamos de processamento em paralelo.

Sintaxe: **ATTACHEP=entryname**

ATTACHEPLOC=entryname address

ATTACHDE=listname address

Onde:

entryname	=> nome do módulo que ganhará o controle
entryname address	=> endereço do módulo que ganhará o controle
listname address	=> endereço do módulo obtido através da chamada anterior da macro BLDL

Exemplo:

ATTACHEP=CICS002

PROGRAMAS

PASSAGEM DE PARÂMETROS

- **ATRAVÉS DO JCL**
  //STEPNAME EXEC PGM=TESTE,PARM='...até 100 bytes...'

- **RECEBENDO NO PROGRAMA**
  Registrador 1 => FULLWORD => TAMANHO PARM(HALFWORD) +
  PARM

  Exemplo:

  ```
  //STEP01   EXEC    PGM=TESTE,PARM='CICP0001'
  ```

  No programa:

  ```
                     L         1,0(0,1)
                     LH        5,0(1)
                     BCTR      5,0
                     EX        MOVEPRM
                     ...
  MOVEPRM            MVC       PARMREC,2(1)
  PARMREC            DC        100C' '
  ```

  Após:

  PARMREC='CICP0001' + 92 caracteres branco

PROGRAMAS

ABENDS

- **TÉRMINO ANORMAL DO PROGRAMA**

  ABENDS DO SISTEMA: SXXX
  ABENDS DO USUÁRIO: UXXX (obtidos pela chamada da macro ABEND)

  ABENDS de sistema mais comuns:

  S0C1 - código de instrução inválido
  S0C4 - endereço de memória não autorizado ou não existe
  S0C7 - formato dos dados incompatível (decimal compactado)
  S0C9 - divisão por zero

  ANÁLISE DE ABENDS:

  - **PSW** - endereço da próxima instrução
  - **OFFSET** - endereço dentro do programa onde ocorreu o ABEND
  - **REGISTRADORES** - conteúdo dos registradores no momento do ABEND
  - **DATA AT PSW** - instruções que estavam sendo executadas
  - **ADDRESS** - endereço do programa na memória virtual
  - **ACTIVE LOAD MODULE** - nome do módulo que estava sendo executado

PROGRAMAS

ABENDS

- **EXEMPLO**

```
SYSTEM COMPLETION CODE=0C1 REASON CODE=00000001
TIME=10.56.12 SEQ=21771 CPU=0000 ASID=0037
   PSW AT TIME OF ERROR 078D0000 800078B2 ILC 2 INTC 01
  ACTIVE LOAD MODULE ADDRESS=00007870 OFFSET=00000042
  NAME=CICBATCH
    DATA AT PSW 000078AC - 4450B078 00000000 D213B059
    GR     0: 00000006_FD000019    1: 00000000_80006FE6
                   2: 00000000_00000040    3: 00000000_009D99D4
           4: 00000000_009D99B0    5: 00000000_00000010
           6: 00000000_00006F48    7: 00000000_FD000000
           8: 00000000_009D72B0    9: 00000000_009D1CD0
           A: 00000000_00000000    B: 00000000_80007870
           C: 00000000_895EFCFA    D: 00000000_00007BFC
           E: 00000000_80FDC2B8    F: 00000000_80007870
END OF SYMPTOM DUMP
```