1 – Software e Linguagens de Programação

1.1 - Software

É um conjunto de programas, rotinas e procedimentos envolvidos na operação de um computador e HARDWARE é o equipamento em si, toda a parte física do computador e seus periféricos (memória, vídeo, teclado, CPU, etc). Aparentemente, a parte principal de um sistema computacional é o hardware, porém esta impressão é desmistificada a medida que se observa a história da computação. Cada vez mais o hardware é considerado como artigo primário, que pode ser fornecido por qualquer fabricante enquanto o software é o mecanismo que nos possibilita dar vazão ao potencial do hardware.

A fim de evidenciar a evolução do software, costuma-se classifica-lo em eras.

A Primeira Era, aproximadamente entre 1950 e 1965 serviu para dar utilidade aos primeiros sistemas computacionais. A programação era uma arte secundária e o desenvolvimento do software feito sem administração. A maior parte do hardware dedicava-se a execução de um único programa que, por sua vez, dedicava-se a uma aplicação específica. Apesar do *hardware* ser de propósito geral, o software era projetado sob medida para cada aplicação e tinha distribuição limitada. A maior parte do software era desenvolvida e utilizada pela própria pessoa ou organização. Processo de fazer, testar e corrigir. Caracterizava-se pela ausência de documentação.

A Segunda Era da evolução dos sistemas vai de 1965 a 1975. Caracteriza-se pela multiprogramação e os sistemas multiusuários. Operação em tempo real e gerenciamento de banco de dados puderam ser implantados devido ao crescente poder do hardware. O software passou a ser encarado como um produto e surgiram os primeiros fabricantes, desenvolvendo os primeiros pacotes. Problemas de manutenção no software – falhas, alterações, personalização – começaram a absorver recursos consideráveis. Verificou-se cada vez mais que software era tarefa mais de engenharia que apenas de programador.

A Terceira Era começa por volta de 1975 e vai até quase o fim da década de 80. Os sistemas distribuídos aumentaram intensamente a complexidade. As redes locais e globais de comunicação digital exigiram esforço considerável dos desenvolvedores de software. Caracterizou-se ainda pelo advento dos microprocessadores nos computadores pessoais (1981) e estações de trabalho (*workstation*). Isso gerou aplicações específicas em um amplo conjunto de produtos inteligentes (carros, forno de microondas, robôs industriais). Esta era impulsionou definitivamente as empresas de software, que chegavam a vender dezenas e até mesmo centenas de milhares de cópias de um software. Enquanto a taxa de crescimento de vendas de computadores estabilizou-se no meio da década de 80, as de software mantiveram-se sempre crescendo. Na indústria ou mesmo em casa, as pessoas gastavam mais dinheiro com software que com o hardware.

A quarta geração começou no meio da década de 80 e se estende até a atualidade. Caracteriza-se para as tecnologias orientadas à objeto, sistemas especialistas, computação paralela e uso de redes neurais artificiais.

1.2 - SISTEMA OPERACIONAL (SO)

É o software que gerencia todo o relacionamento entre as unidades componentes do equipamento. Ele quem distribui os recursos necessários no momento certo, como se fosse um gerente, que precisa ser tão mais 'sofisticado' e 'competente' quanto maiores e mais diversificados forem o tamanho e a natureza dos recursos disponíveis.

É por este motivo que um microcomputador com apenas um teclado necessitará de um sistema operacional bem mais modesto que um computador de grande porte (normalmente ligado a um grande número de periféricos de entrada e saída, como por exemplo, um elevado número de terminais espalhados em pontos distantes).

Um usuário, ao submeter um trabalho ao computador estará provocando a execução de um conjunto enorme de atividades. Algumas tarefas desenvolvidas pelo SO são:

· Lê e responde a comandos do usuário;
· Permite o armazenamento de programas do usuário;
· Coordena o fluxo de informação entre as memórias auxiliares e os diferentes dispositivos de entrada e saída;
· Coloca as instruções e os dados na memória e executa-os ordenadamente;
· Realiza tarefas de distribuição de tempo do sistema;
· Controla as interrupções do sistema;
· Controle da situação do sistema e envio de mensagens informativas ao usuário.
· Faz a gestão dos diferentes programas em execução
· Ocupa-se da interface entre programas e usuários

O SO é quem define o ambiente no qual você interage com o computador. Estabelece o padrão funcional do computador, definindo muitos dos limites práticos da utilização dele, assim como o hardware específico o faz. Ele completa seu computador, dando-lhe vida e características particulares. É importante salientar que os programas não trabalham com qualquer sistema operacional, sendo específicos para cada um deles. Os primeiros SO (por ex, o MS-DOS *(Microsoft Disk-Operating Syste*m)) usavam comandos, que o usuário tinha que memorizar. Gradualmente, os ícones substituíram os comandos, dando origem às interfaces gráficas (GUIs).

Programas podem ser construídos diretamente no *hardwar*e, como em aplicações especificas. (calculadoras, microondas).

Em alguns computadores especializados, o conjunto de instruções é incorporado no circuito. Por exemplo, calculadoras, relógios de pulso, motores de automóveis, fornos microondas, motores de automóveis, etc.

O SO deve ser capaz de gerenciar as quatro unidades principais, que fazem parte de um sistema operacional típico:

1.2.1 - MEMÓRIA

Áreas específicas de um computador onde ficam armazenadas as instruções de programas a serem executadas pelo processador e também os dados. O gerente de memória precisa saber, a cada instante, quais áreas estão ocupadas e quais estão livres para poder utilizá-las corretamente, armazenando os dados e instruções necessárias em locais (endereços) específicos e pré-selecionados.

Via de regra, um programa é formado por diversas linhas de instruções que por sua vez são formadas por diversos caracteres. Como cada caracter ocupa uma posição de memória, na maioria das vezes necessitaremos de milhares destas posições. Para quantificá-las usamos múltiplos de 1.024 Bytes, chamados de Kilobytes (KB), ou de 1.048.576 Bytes, chamados de Megabytes (MB).

1.2.2 - PROCESSAMENTO

O gerente de processamento (CPU), controla a execução de tarefas (necessárias a cada trabalho) a partir de um plano (programa) para sua execução dentro da máquina.

1.2.3 - DISPOSITIVOS

O gerente dos dispositivos administra o tráfego de entrada e saída entre os periféricos e a CPU.

1.2.4 - DADOS

O gerente de dados, gerencia os arquivos que estejam na máquina, abrindo-os e fechando-os no início e fim de cada processamento, assim como acompanhando o percurso deles ao longo de todas as atitudes.

1.3 - RECURSOS DE SOFTWARE

Seria complicada a utilização de um computador se somente tivéssemos o *hardware* e o sistema operacional, sendo por esse motivo que bem próximo ao SO também encontramos as chamadas linguagens ou programas de apoio, como:

· Assembler;

· Compiladores de linguagem;

· Programas utilitários ou de serviços;

· Programas aplicativos.

1.3.1 - ASSEMBLY

É uma linguagem básica estrutural característica de cada equipamento. O programa Assembler é quem traduz o código interno de instruções da máquina. Os programas de apoio e inclusive partes importantes do sistema operacional são construídos, na maioria dos casos, a partir do mesmo.

1.3.2 - COMPILADORES

Funcionam como tradutores de uma linguagem de uso mais compreensível pelo homem (linguagem de Alto Nível), como Cobol, Fortran, PL-1, etc., para linguagem de máquina (realmente entendida pela máquina). Podem ser de diferentes tipos:

· Interpretadores, interpretam instrução a instrução; são mais lentos do que os programas escritos diretamente em linguagem máquina.

· Compiladores, traduzem todo o código num novo código; executam com a mesma rapidez que os programas escritos diretamente em linguagem máquina.

· Assemblers, interpretam a linguagem simbólica em linguagem máquina.

1.3.3 - PROGRAMAS UTILITÁRIOS OU DE SERVIÇOS

Estes programas, normalmente fornecidos pelos próprios fabricantes do equipamento, são de fácil execução e uso. Ex: copiar um arquivo de um disco para outro, apagar (deletar) um arquivo, classificar um arquivo, são tarefas executadas pelos programas utilitários.

1.3.4 - PROGRAMAS APLICATIVOS

São programas normalmente complexos e extensos que buscam apoiar totalmente os usuários em tarefas como: Editar Textos, controlar Banco de Dados, controlar uma Planilha, fazer planejamentos. Os programas gerados com fins específicos, pelo próprio usuário ou não, também são programas aplicativos.

1.4 – Algumas Linguagens de Programação

As linguagens de programação representam a interface entre o programador e o hardware, ou a ferramenta utilizada para materializar uma tarefa que será desenvolvida pelo sistema. Cada linguagem tem características próprias e foi concebida e otimizada para atuar em áreas especificas.

Na base de todas encontra-se a linguagem de baixo nível, mais voltada ao hardware e que portanto, exige conhecimento considerável do hardware que receberá o programa. Na base esta a linguagem de máquina que é a versão final que o computador realmente processa. Há ainda a linguagem Assembly, intimamente ligada ao conjunto de instruções do processador.

Num nível intermediário encontram-se as linguagens que apresentam facilidade de programação com comandos, porém conseguem amplo acesso ao hardware, como as linguagens de baixo nível. São ideais para elaboração de programas como sistemas operacionais ou rotinas que interagem com dispositivos de I/O, memórias, periféricos. São as chamadas linguagens de médio nível e como exemplo tem-se a Linguagem C.

O terceiro tipo de linguagem são as chamadas linguagens de alto nível e que oferecem maior facilidade de programação. Este conceito já é ultrapassado, pois o desenvolvimento de pacotes permite ampla utilização de linguagens de todos os níveis de forma simples pelo usuário. Alguns softwares permitem até que se misturem blocos com diferentes linguagens (ex. Delphi e Assembly). Outro tipo de ambiente, voltado a objetos, estabelece mais um nível às linguagens, onde as antigas linhas de código sequer aparecem. Segue um breve resumo das linguagens mais conhecidas.

· A IBM desenvolveu uma linguagem que visava o uso de computadores em aplicações científicas. Desenvolvida entre 1954 e 1957, o FORTRAN *(FORmula TRANslato*r) foi a primeira linguagem de alto nível a ser largamente usada.

Em 1957, a *Association for Computing Machinery* decidiu desenvolver uma linguagem que corrige algumas das falhas encontradas no FORTRAN. Um ano mais tarde aparece o ALGOL *(ALGOrithmic Languag*e), outra linguagem com orientação científica. Largamente usada na Europa entre 1960 e 1970, o ALGOL foi depois substituída por outras linguagens enquanto que o FORTRAN continuou a ser usado.

A principal razão para a continuação do uso do Fortran foi o enorme investimento em programas.

· **COBOL** *(COmmon Business Oriented Languag*e), uma linguagem de programação comercializável e largamente usada nos negócios, concentrava-se na organização de dados e manuseio de fichas. Ainda utilizada hoje em dia.

· **BASIC** *(Beginner's All-purpose Symbolic Instruction Code)* foi desenvolvida no Dartmouth College no início dos anos 60 visando hobbistas. O seu uso tornou-se universal com a popularização do microcomputador entre 1970 e 1980. Apesar de ser lenta e ineficiente era simples de aprender. Uma das razões para a sua enorme divulgação foi o fato de muitos computadores serem vendidos com instruções incorporadas no *hardware* (memória ROM) escritas em BASIC.

· **PASCAL**, originalmente projetada como uma linguagem pedagógica, é hoje em dia ainda uma linguagem da programação popular.

· **LOGO** foi desenvolvida com a intenção de ensinar as crianças a usarem computadores.

· **C**, uma linguagem desenvolvida pelos laboratórios Bell em 1970, é largamente usada no desenvolvimento de *software* de sistemas, por exemplo, na escrita de compiladores.

· **LISP e PROLOG** são largamente usadas em inteligência artificial.

· **Java** – Desenvolvida pela SUN é uma linguagem de propósito geral, orientada a objetos, especificamente projetada para ter pouca dependência do hardware. Os programas desenvolvidos podem ser executados através da internet.

2 – Tabela de Instruções e Código de Linguagem de Máquina do 8085

THE INSTRUCTION SET

8085A CPU INSTRUCTIONS IN OPERATION CODE SEQUENCE
Table 5-2

OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC
00	NOP	2B	DCX H	56	MOV D,M	81	ADD C	AC	XRA H	D7	RST 2
01	LXI B,D16	2C	INR L	57	MOV D,A	82	ADD D	AD	XRA L	D8	RC
02	STAX B	2D	DCR L	58	MOV E,B	83	ADD E	AE	XRA M	D9	–
03	INX B	2E	MVI L,D8	59	MOV E,C	84	ADD H	AF	XRA A	DA	JC Adr
04	INR B	2F	CMA	5A	MOV E,D	85	ADD L	B0	ORA B	DB	IN D8
05	DCR B	30	SIM	5B	MOV E,E	86	ADD M	B1	ORA C	DC	CC Adr
06	MVI B,D8	31	LXI SP,D16	5C	MOV E,H	87	ADD A	B2	ORA D	DD	–
07	RLC	32	STA Adr	5D	MOV E,L	88	ADC B	B3	ORA E	DE	SBI D8
08	–	33	INX SP	5E	MOV E,M	89	ADC C	B4	ORA H	DF	RST 3
09	DAD B	34	INR M	5F	MOV E,A	8A	ADC D	B5	ORA L	E0	RPO
0A	LDAX B	35	DCR M	60	MOV H,B	8B	ADC E	B6	ORA M	E1	POP H
0B	DCX B	36	MVI M,D8	61	MOV H,C	8C	ADC H	B7	ORA A	E2	JPO Adr
0C	INR C	37	STC	62	MOV H,D	8D	ADC L	B8	CMP B	E3	XTHL
0D	DCR C	38	–	63	MOV H,E	8E	ADC M	B9	CMP C	E4	CPO Adr
0E	MVI C,D8	39	DAD SP	64	MOV H,H	8F	ADC A	BA	CMP D	E5	PUSH H
0F	RRC	3A	LDA Adr	65	MOV H,L	90	SUB B	BB	CMP E	E6	ANI D8
10	–	3B	DCX SP	66	MOV H,M	91	SUB C	BC	CMP H	E7	RST 4
11	LXI D,D16	3C	INR A	67	MOV H,A	92	SUB D	BD	CMP L	E8	RPE
12	STAX D	3D	DCR A	68	MOV L,B	93	SUB E	BE	CMP M	E9	PCHL
13	INX D	3E	MVI A,D8	69	MOV L,C	94	SUB H	BF	CMP A	EA	JPE Adr
14	INR D	3F	CMC	6A	MOV L,D	95	SUB L	C0	RNZ	EB	XCHG
15	DCR D	40	MOV B,B	6B	MOV L,E	96	SUB M	C1	POP B	EC	CPE Adr
16	MVI D,D8	41	MOV B,C	6C	MOV L,H	97	SUB A	C2	JNZ Adr	ED	–
17	RAL	42	MOV B,D	6D	MOV L,L	98	SBB B	C3	JMP Adr	EE	XRI D8
18	–	43	MOV B,E	6E	MOV L,M	99	SBB C	C4	CNZ Adr	EF	RST 5
19	DAD D	44	MOV B,H	6F	MOV L,A	9A	SBB D	C5	PUSH B	F0	RP
1A	LDAX D	45	MOV B,L	70	MOV M,B	9B	SBB E	C6	ADI D8	F1	POP PSW
1B	DCX D	46	MOV B,M	71	MOV M,C	9C	SBB H	C7	RST 0	F2	JP Adr
1C	INR E	47	MOV B,A	72	MOV M,D	9D	SBB L	C8	RZ	F3	DI
1D	DCR E	48	MOV C,B	73	MOV M,E	9E	SBB M	C9	RET Adr	F4	CP Adr
1E	MVI E,D8	49	MOV C,C	74	MOV M,H	9F	SBB A	CA	JZ	F5	PUSH PSW
1F	RAR	4A	MOV C,D	75	MOV M,L	A0	ANA B	CB	–	F6	ORI D8
20	RIM	4B	MOV C,E	76	HLT	A1	ANA C	CC	CZ Adr	F7	RST 6
21	LXI H,D16	4C	MOV C,H	77	MOV M,A	A2	ANA D	CD	CALL Adr	F8	RM
22	SHLD Adr	4D	MOV C,L	78	MOV A,B	A3	ANA E	CE	ACI D8	F9	SPHL
23	INX H	4E	MOV C,M	79	MOV A,C	A4	ANA H	CF	RST 1	FA	JM Adr
24	INR H	4F	MOV C,A	7A	MOV A,D	A5	ANA L	D0	RNC	FB	EI
25	DCR H	50	MOV D,B	7B	MOV A,E	A6	ANA M	D1	POP D	FC	CM Adr
26	MVI H,D8	51	MOV D,C	7C	MOV A,H	A7	ANA A	D2	JNC Adr	FD	–
27	DAA	52	MOV D,D	7D	MOV A,L	A8	XRA B	D3	OUT D8	FE	CPI D8
28	–	53	MOV D,E	7E	MOV A,M	A9	XRA C	D4	CNC Adr	FF	RST 7
29	DAD H	54	MOV D,H	7F	MOV A,A	AA	XRA D	D5	PUSH D		
2A	LHLD Adr	55	MOV D,L	80	ADD B	AB	XRA E	D6	SUI D8		

D8 = constant, or logical/arithmetic expression that evaluates to an 8-bit data quantity.

D16 = constant, or logical/arithmetic expression that evaluates to a 16-bit data quantity.

Adr = 16-bit address.

2 – Conjunto de Instruções do 8085

2.1 – Registradores Microprocessador 8085

O 8085 possui 7 registradores de uso geral, todos de 8 bits, organizados como segue:

8 bits	**8 bits**
Acc	
B	**C**
D	**E**
H	L

O Acc, ou simplesmente A, é denominado de ACUMULADOR. Esse registrador (REG) é utilizado em todas as operações de entrada/saída e em quase todas as instruções de lógica e aritmética. Os outros registradores são também muito importantes, principalmente porque podem ser utilizados em pares, formando registrados de 16 bits, organizados como: par BC, DE e HL.

Existem outros registradores no 8085 que podem ser usados em algumas operações e que são indispensáveis à sua operação, como o apontador de pilha (SP), o contador de programa (PC), o registrador de instruções e o registrador para FLAGS

O 8085 trabalha com 8 bits de dados (1 byte) e 16 linhas de endereço. Assim, um importante recurso é utilizar pares de registradores para endereçamento indireto de memória.

2.2 - Conjunto de Instruções do 8085 e Classificação

A operação de qualquer microprocessador é baseada em um conjunto de códigos que é denominado de CONJUNTO DE INSTRUÇÕES. Esses códigos deverão ser inseridos necessariamente em uma memória, e em muitos casos dados adicionais são necessários. Desta forma, o microprocessador pode apresentar diferentes modos para acesso à memória, chamados de *modos de endereçamento*.

As instruções no 8085 são organizadas como segue:

1^o byte $\rightarrow$ código (sempre)

2^o byte $\rightarrow$ dado ou byte menos significativo de endereço(LSB)

3^o byte $\rightarrow$ dado ou byte mais significativo de endereço(MSB)

O conjunto de instruções do 8085 está organizado em 5 grupos básicos:

1 - transferência de dados, que permitem a troca de dados entre registradores e memória;

2 – operações lógicas, que permitem a implementação de funções lógicas, como AND, OR, XOR, comparações, etc.;

3 – operações aritméticas, que permitem a realização de somas e subtrações;

4 – controle de programa, que permitem desvios condicionados ou não, chamadas de subrotinas, etc.

5 – manipulação de PILHA e acesso aos dispositivos de entrada/saída.

2.3 - Modos de Endereçamento no 8085

Pode-se classificar as instrução do 8085 de acordo com os seguintes modos de endereçamento:

→ implícito: o microprocessador "sabe" a priori onde está o dado (ex.: CMA, DAA, etc.)

→ por registro: operações envolvendo registradores internos (ex.: ADD B, SUB D, etc)

→ imediato: o dado segue o código da instrução (ex.: MVI A,25; CPI 00, etc)

→ direto: o endereço do dado segue o código (ex.: STA 2550, LDA 4001, etc)

→ indireto: o endereço do dado é passado via um par de registros (ex.: LDAX B, etc)

2.4 – Instruções de Transferência de Dados do 8085

As instruções deste grupo realizam as seguintes funções:

- transferência de dados entre registradores e entre registrador e memória;
- atribuir valor ao registrador ou à memória;
- transferência de dados entre acumulador e memória;
- transferência de dados entre os registradores H e L e a memória;
- troca de valores entre o par de registradores H,L e o par D,E.

Antes de apresentar as instruções, considere as seguintes definições:

- r, r1, r2 : registradores - A, B, C, D, E, H, L;
- M : símbolo indicando endereço de memória formado pelo conteúdo do par H, L;
- Data8 – Valor de 8 bits
- Data16 – Valor de 16 bits (dado)
- Adr16 – Valor de 16 bits (endereço)
- DDD – registrador de destino do byte (dados)
- FFF – registrador fonte do byte (dados)

Registrador	Composição de Bits DDD ou FFF
B	**000**
C	**001**
D	**010**
E	**011**
H	**100**
L	**101**
A	**111**

- **PR : Par de registradores**

NOME	REGISTRADORES	Composição de Bits PR
B	**B, C**	**00**
D	**D, E**	**01**
H	**H, L**	**10**
SP	**SP**	**11**

- **rh : registrador superior (B, D e H);**
- **rl : registrador inferior (C, E, L);**
- **Tempo de Execução ou Processamento de uma instrução:**

$$T_E = \frac{\text{Número de Estados T(T)}}{f_{clock}} = \frac{T}{2.35*10^6} = 425.10^{-9} * T \qquad [s]$$

1 - **MOV r1, r2** (r1) ← (r2) **01 DDD FFF** T = 4

Move

O conteúdo do registrador r2 é copiado no registrador r1.

Exemplos:

Instrução	Código (Binário)	Código (Hexadecimal)
MOV A, A	01 111 111	7F
MOV C, D	01 001 010	4A
MOV H, B	01 100 000	60

2 - MOV r, M (r) ← ((H)(L)) **01 DDD 110** T = 7

Move

O conteúdo do endereço de memória formado pelo par H,L é copiado no registrador r.

Exemplos:

Instrução	Código (Binário)	Código (Hexadecimal)
MOV A, M	01 111 110	7E
MOV C, M	01 001 110	4E

3 - MOV M, r ((H)(L)) ← (r) **01110 FFF** T = 7

Move

O conteúdo do registrador r é copiado no endereço de memória formado pelo par H,L.

Exemplos:

Instrução	Código (Binário)	Código (Hexadecimal)
MOV M, A	0111 0 111	77
MOV M, H	0111 0 100	74

4 - MVI r, Data8 (r) ← Data8 00 DDD 110 Data8 T = 7

Move Imediate

O byte Data 8 é transferido para o registrador r.

Exemplos:

Instrução	Código (Binário)	Código (Hexadecimal)
MVI A, 45H	00 111 110 0100 0101	3E 45
MVI B, 06H	00 000 110 0000 0110	06 06

5 - **MVI M, Data8** ((H)(L)) ← Data8 0011 0110 Data8 T = 10

Move Imediate

O byte Data 8 é transferido para o endereço de memória dado pelo par HL.

Exemplo:

Instrução	Código (Binário)	Código (Hexadecimal)
MVI M, DCH	0011 0110	36

	1101 1100	DC
ANTES (H) = 20H (L) = 30H (2030H) = XX		APÓS (H) = 20H (L) = 30H (2030H) = DC H

6 - LXI PR, Data16 (rh) ← $Data16_H$ 00 PR 0001 T = 10
Load Extended Imediate (rl) ← $Data16_L$ $Data16_L$
$Data16_H$

O byte superior de Data16 é transferido para o registrador superior rh e o byte inferior de Data16 é transferido para o registrador inferior rl.

Exemplo:

Instrução	Código (Binário)	Código (Hexadecimal)
LXI D, 20B0H	00 01 0001 1011 0000 0010 0000	11 B0 20
ANTES (D) = XX (E) = XX		APÓS (D) = 20H (E) = B0H

7 - LDA Adr16 (A) ← (Adr16) 0011 1010 T = 13
Load Accumulator $Adr16_L$
$Adr16_H$

O conteúdo do endereço Adr16 é copiado no acumulador.
Exemplo:

Instrução	Código (Binário)	Código (Hexadecimal)
LDA 20B0H	0011 1010 1011 0000 0010 0000	3A B0 20
ANTES (20B0H) = CD H (A) = XX		APÓS (20B0H) = CD H (A) = CD H

8 - STA Adr16 (Adr16) ← (A) 0011 0010 T = 13
Store Accumulator $Adr16_L$
$Adr16_H$

O conteúdo do acumulador é copiado no endereço Adr16.
Exemplo:

Instrução	Código (Binário)	Código (Hexadecimal)
STA 20B0H	0011 0010 1011 0000 0010 0000	32 B0 20
ANTES (20B0H) = XX (A) = CDH		APÓS (20B0H) = CDH (A) = CDH

9 - LHLD Adr16 (L) ← (Adr16) 0010 1010 T = 16
Load HL Direct (H) ←(Adr16+1) $Adr16_L$
$Adr16_H$

O conteúdo do endereço Adr16 é copiado no registrador L e o conteúdo do endereço seguinte, Adr16+1, é copiado no registrador H.

Exemplo:

Instrução	Código (Binário)	Código (Hexadecimal)
LHLD 20B0H	0010 1010	2A

	1011 0000 0010 0000	B0 20
ANTES (20B0H) = 23 H (20B1H) = 45 H (H) = XX (L) = XX		APÓS (20B0H) = 23H (20B1H) = 45H (H) = 45H (L) = 23H

10 - SHLD Adr16 (Adr16) ← (L) 0010 0010 T = 16
Store HL Direct (Adr16+1) ←(H) $Adr16_L$
$Adr16_H$

O conteúdo do registrador L é copiado no endereço Adr16 e o conteúdo registrador H e copiado no endereço seguinte, Adr16+1.

Exemplo:

Instrução	Código (Binário)	Código (Hexadecimal)
SHLD 2035H	0010 0010 0011 0101 0010 0000	22 35 20
ANTES (2035H) = XX (2036H) = XX (H) = 31 H (L) = F4 H		APÓS (2035 H) = F4 H (2036H) = 31 H (H) = 31 H (L) = F4 H

11 - LDAX PR (A) ← ((PR)) 00 SR 1010 T = 7
Load Accum. Extended PR só B e D

O conteúdo do endereço formado pelo par de registradores PR é copiado no Acumulador.

Exemplo:

Instrução	Código (Binário)	Código (Hexadecimal)
LDAX B	00 00 1010	0A
ANTES (B) = 20H (C) = 30H (2030H) = 5FH (A) = XX		APÓS (B) = 20H (C)= 30H (2030H) = 5FH (A) = 5FH

12 - STAX PR ((PR)) ← (A) 00 SR 0010 T = 7
Store Accum. Extended PR só B e D

O conteúdo do Acumulador e copiado no endereço formado pelo par de registradores PR.

Exemplo:

Instrução	Código (Binário)	Código (Hexadecimal)
STAX D	00 01 0010	12
ANTES (D) = 20H (E) = 30H (2030H) = XX H (A)= C3H		APÓS (D) = 20H (E) = 30H (2030H) = C3H (A) = C3H

13 - XCHG (H) ↔ (D) 1110 1011 T = 4
Exchange (L) ↔ (E)

O conteúdo dos registradores H e D e os registradores L e E são trocados.

Exemplo:

Instrução	Código (Binário)	Código (Hexadecimal)
XCHG	1110 1011	EB

ANTES	APÓS
(D) = 20H (E) = 30H (H) = 45H (L) = 13H	(D) = 45H (E) = 13H (H) = 20H (L) = 30H