Exemplo: como devemos programar o SAP-1 para resolver este problema de aritmética? 16 + 29 + 24 - 32
Armazenar o os dados (6, 20, 4, 32) nos locais da memória 9H a CH. Converter os dados decimais em hexadecimais (terceira tabela).

Endereço	Conteúdo
0H	LDA 9H
1H	ADD AH
2H	ADD BH
3H	SUB CH
4H	OUT
5H	HLT
6H	XX
7H	XX
8H	XX
9H	10H
AH	14H
BH	18H
CH	20H

A versão em linguagem de máquina é

Endereço	Conteúdo
0000	0000 1001
0001	00011010
0010	00011011
0011	0010 1100
0100	1110 XXXX
0101	1111 XXXX
0110	XXXX XXXX
0111	XXXX XXXX
1000	XXXX XXXX
1001	0001 0000
1010	0001 0100
1011	0001 1000
1100	0010 0000

Endereço	Conteúdo
0H	09H
1H	1AH
2H	1BH
3H	2CH
4H	EXH
5H	FXH
6H	XXH
7H	XXH
8H	XXH
9H	10H
AH	14H
BH	18H
CH	20H

O programa está na memória inferior e os dados na memória superior. O PC aponta para o endereço 0000 para a primeira instrução, 0001 para a segunda instrução e assim por diante. Podemos compactar o programa e os dados do exemplo precedente convertendo em abreviação hexadecimal.

Ciclo de máquina: cada ciclo de máquina dura exatamente seis estados T, não importa qual a instrução. É necessário um ciclo de máquina para buscar e executar cada instrução.
Ciclo de busca (FETCH). Os 3 primeiros estados sempre são esses:
T1: estado de endereço. O endereço no PC é transferido para o registrador de endereços de memória (REM) durante este estado. Durante o estado de endereço, EP e /Lm estão ativos; todos os outros bits de controle estão inativos. Isto significa que o controlador-sequencializador está enviando a seguinte palavra de controle:
T2: estado de incremento. O contador de programa é incrementado. O bit Cp é ativo e os demais inativos.
T3: estado de memória. A instrução de RAM endereçada é transferida da memória para o registrador de instrução. Os bits de controle ativos são /Ce e /Li.

$\mathbf{CON}= C_P E_P \overline{L_M}\,\overline{CE}\ \overline{L_I}\,\overline{E_I}\,\overline{L_A} E_A\ S_U E_U \overline{L_B}\,\overline{L_0}$
$= 0\ 1\ 0\ 1\ \ 1\ 1\ 1\ 0\ \ 0\ 0\ 1\ 1$

$\mathbf{CON}= C_P E_P \overline{L_M}\,\overline{CE}\ \overline{L_I}\,\overline{E_I}\,\overline{L_A} E_A\ S_U E_U \overline{L_B}\,\overline{L_0}$
$= 1\ 0\ 1\ 1\ \ 1\ 1\ 1\ 0\ \ 0\ 0\ 1\ 1$

$\mathbf{CON}= C_P E_P \overline{L_M}\,\overline{CE}\ \overline{L_I}\,\overline{E_I}\,\overline{L_A} E_A\ S_U E_U \overline{L_B}\,\overline{L_0}$
$= 0\ 0\ 1\ 0\ \ 0\ 1\ 1\ 0\ \ 0\ 0\ 1\ 1$

Ciclo de execução. Rotinas das principais operações:
LDA: **T4**= ativamos /Ei e /Lm. O campo de instrução vai para o c/s, onde ele é decodificado e o campo de endereços é carregado no REM. **T5**= ativamos /Ce e /La. A palavra de dados endereçada a RAM será carregada no acumulador na próxima transição positiva de relógio. **T6**= NOP (o c/s envia para fora uma palavra em que todos os bits estão inativos).
ADD: **T4**= /Ei e /Lm. O campo de instruções vai para o controlador-sequencializador e o campo de endereços para o REM. **T5**= /Ce e /Lb. Permite que a palavra de RAM endereçada prepare (sete) o registrador B. **T6**= Eu e /La. O somador-subtrator estabelece ou prepara o acumulador. A transição positiva de relógio carrega a soma no acumulador.
SUB: **T4**= /Ei e /Lm. **T5**= /Ce e /Lb. **T6**= Eu, /La e Su. Quando Su é alto, o complemento de 1 é transmitido e um 1 é acrescentado ao LSB para formar o complemento de 2.
OUT (IR = 1110 XXXX): **T4**= Ea e /L0. Transfere a palavra do acumulador para o registrador de saída quando ocorre a borda do CLK. **T5** e **T6** = NOP
HLT (IR = 1111 XXXX): não requer uma rotina de controle porque não há registradores envolvidos na execução de uma instrução HLT. O campo 1111 de instrução avisará ao c/s para interromper o processamento dos dados. O c/s para o computador desligando o relógio.

Microprogramação: armazenar microinstruções em uma ROM. Estas microinst. podem ser armazenadas em uma ROM de controle com a rotina de busca nos endereços 0H a 2H, a rotina LDA nos endereços 3H a 5H, a rotina ADD em 6H a 8H, a rotina SUB em 9H e BH e a rotina OUT em CH a EH. Para ter acesso a qualquer rotina, precisamos fornecer os endereços corretos. Por exemplo, para obter a rotina ADD, necessitamos fornecer os endereços 6H, 7H e 8H. Portanto, ter acesso a qualquer rotina requer 3etapas: 1. Conhecimento do endereço de partida da rotina. 2. Escalonamento através dos endereços da rotina. 3. Aplicação dos endereços à ROM de controle.
As instruções LDA, ADD, SUB, ... são chamadas macroinstruções. Cada macroinst. do SAP-1 é formada de três microinst. Quando T3 for alto, o contador pré-ajustável carregará o endereço de partida da ROM de endereços. Durante os outros estados T, o contador contará. Quando começa o processamento no computador, a saída do contador por padrão é 0000 durante o estado T1, 0001 durante o T2 e 00 10 durante T3 . Cada ciclo de busca é o mesmo, porque 0000, 0001 e 0010 saem do contador durante os estados T1, T2 e T3.
O OPCODE no IR controla o ciclo de execução. Se uma instrução ADD tiver sido buscada, os bits I7I6I5I4 serão 0001. Estes bits do código op comandam a ROM de endereços, produzindo uma saída de 0110. Este endereço de partida é a entrada para o contador pré-ajustável. Quando T3 estiver alto, a próxima transição negativa de relógio carregará 0110 no contador pré-ajustável. O contador está agora ajustado (levado-a-1), e a contagem pode resumir-se no endereço de partida da rotina ADD. A saída do contador é 0110 durante o estado T4,0111 durante o estado T5 e 1000 durante o estado T6.
Vantagem: é muito mais fácil armazenar microinstruções em uma ROM do que montar um decodificador de instruções e matriz de controle (uma vez montados, a única maneira pela qual podemos alterar o conjunto de instruções é desconectando e montando novamente). Já na ROM tudo o que temos a fazer é modificar a ROM de controle e a ROM de endereço de partida.A ROM de endereços dentro do c/s contém esses dados:

OPERAÇÃO: Antes de cada processamento no computador, o operador introduz o programa e os dados na memória do SAP-1. Cada ciclo de máquina do SAP-1 começa com um ciclo de busca (no fim dele a instrução é armazenada no registrador de instruções). Depois do campo de instrução ter sido decodificado, o c/s automaticamente gerará a rotina de execução correta. Ao término do ciclo de execução, o contador em anel do PC se restabelecerá (é zerado) e começará o próximo ciclo de máquina. O processamento dos dados terminará quando uma instrução HLT for carregada no registrador de instruções.

INC: **T4** (na pos10H da ROM do c/s)= INC, Eu, /La. **T5** (na pos11H da ROM do c/s) e **T6** (na pos12H da ROM do c/s)= NOP
DEC: **T4** (na pos13H da ROM do c/s)= DEC, Su, Eu, /La. **T5** (na pos14H da ROM do c/s) e **T6** (na pos15H da ROM do c/s)= NOP
JMP: **T4**= /Ei e Lp. **T5** e **T6** = NOP

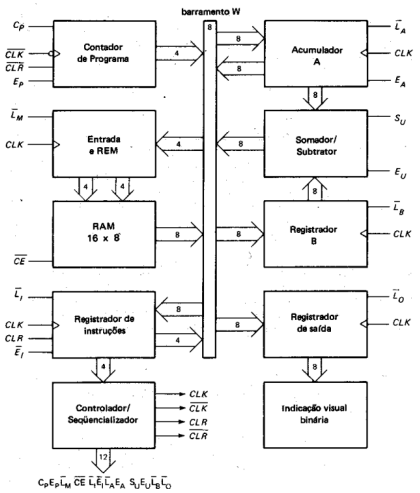
Barramento O SAP-1 utiliza um único barramento principal compartilhado por todos os componentes, cada um habilitando sua saída tri-state quando precisa colocar dados, evitando conflitos.
Contador de Programa (PC) Inicialmente zerado, fornece endereços de instruções armazenadas no início da RAM. Ao habilitar sua saída, o PC coloca o endereço no barramento. Quando não habilitado, pode ser incrementado internamente sem alterar o barramento. No inicio de um processamento de

computador, um /CLR restabelece o contador de programa 0000. No T1, um EP alto coloca o endereço no barramento W. No T2, um CP alto incrementa o contador de programa. O PC é inativo durante os estados T3 a T6 .

Macro	Estado	CON	Ativo
LDA	T_4	1A3H	$\overline{L}_M, \overline{E}_I$
	T_5	2C3H	$\overline{CE}, \overline{L}_A$
	T_6	3E3H	**Nada**
ADD	T_4	1A3H	$\overline{L}_M, \overline{E}_I$
	T_5	2E1H	$\overline{CE}, \overline{L}_B$
	T_6	3C7H	$\overline{L}_A, E_U$
SUB	T_4	1A3H	$\overline{L}_M, \overline{E}_I$
	T_5	2E1H	$\overline{CE}, \overline{L}_B$
	T_6	3CFH	$\overline{L}_A, S_U, E_U$
OUT	T_4	3F2H	$E_A, \overline{L}_0$
	T_5	3E3H	**Nada**
	T_6	3E3H	**Nada**

$\star$ CON= $C_P E_P \overline{L}_M \overline{CE}\ \overline{L}_I \overline{E}_I \overline{L}_A E_A\ S_U E_U \overline{L}_B \overline{L}_O$

Endereço	Conteúdos	Rotina	Ativo
0H	5E3H	Fetch	$E_P, \overline{L}_M$
1H	BE3H		C_P
2H	263H		$\overline{CE}, \overline{L}_A$
3H	1A3H	LDA	$\overline{L}_M, \overline{E}_I$
4H	2C3H		$\overline{CE}, \overline{L}_A$
5H	3E3H		**Nada**
6H	1A3H	ADD	$\overline{L}_M, \overline{E}_I$
7H	2E1H		$\overline{CE}, \overline{L}_B$
8H	3C7H		$\overline{L}_A, E_U$
9H	1A3H	SUB	$\overline{L}_M, \overline{E}_I$
AH	2E1H		$\overline{CE}, \overline{L}_B$
BH	3CFH		$\overline{L}_A, S_U, E_U$
CH	3F2H	OUT	$E_A, \overline{L}_0$
DH	3E3H		**Nada**
EH	3E3H		**Nada**
FH	**X**	**X**	**Não usado**



REM (MAR) Recebe o endereço do barramento e mantém-no estável para a RAM, permitindo que o PC incremente sem perder o endereço atual. Sem a MAR, a RAM veria valores instáveis se o PC alterasse o endereço rapidamente.

RAM Armazena instruções e dados carregados antes da execução. Recebe o endereço da MAR e fornece a instrução ou dado correspondente ao barramento. Assim, o programa e seus dados são acessados conforme o PC avança.

IR (Registrador de Instruções) Carrega a instrução do barramento e a mantém disponível. Sem o IR, após a RAM mudar de endereço, a instrução anterior se perderia. O IR preserva o opcode e o operando até o término da execução daquela instrução.

Controlador-Sequencializador (c/s) Possui estados internos (T0, T1, T2...) que definem a sequência de micro-operações. O IR fornece o opcode, o c/s usa opcode e estado interno para gerar sinais de controle adequados a cada etapa da instrução. No início, o c/s está num estado pré-definido que faz o PC colocar o endereço inicial no barramento e a MAR carregá-lo. Depois avança estado a estado, ajustando sinais conforme a instrução exige.

Acumulador (A) Recebe resultados intermediários. Sua saída sempre alimenta o somador-subtrator e pode, quando habilitada, colocar seu valor no barramento. **Somador-Subtrator** Soma ou subtrai conforme sinal SU e produz resultados imediatos. **Registrador B** Carrega do barramento dados para operações aritméticas com A. **Registrador de Saída** Recebe o valor de A quando solicitado e o apresenta externamente.

Monte o programa abaixo para o Código de Máquina do SAP-1 completando a tabela nos formatos bi determine o resultado da operação realizada.

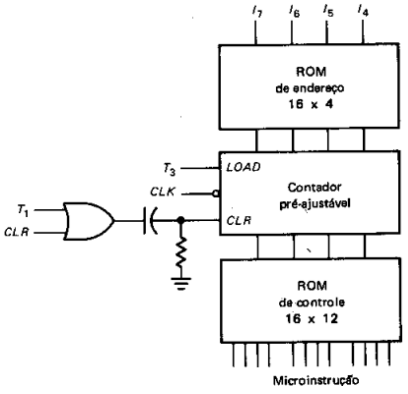
End	Assembly	Cod. Máquina	Cod. Hex	Comentário
0h	LDA Eh	0000 1110	0E	Carrega dado do endereço Eh
1h	ADD Fh	0001 1111	1F	Soma dado do endereço Fh
2h	OUT	1110 0000	E0	Envia para display
3h	ADD Fh	0001 1111	1F	Soma dado do endereço Fh
4h	SUB Ah	0010 1010	2A	Subtrai dado do endereço Ah
5h	OUT	1110 0000	E0	Envia para display
6h	ADD Ch	0001 1100	1C	Soma dado do endereço Ch
7h	OUT	1110 0000	E0	Envia para display
8h	HLT	1111 0000	F0	Termina o programa
9h	–	0000 0000	00	–
Ah	–	1011 1110	–	Dado
Bh	–	0011 1110	–	Dado
Ch	–	0010 1110	–	Dado
Dh	–	1100 0011	–	Dado
Eh	–	1100 0000	–	Dado
Fh	–	0000 1111	–	Dado

DISPLAY:

Resultado em Binário	Resultado em Hexadecimal
1º OUT: 1100 1111	CF
2º OUT: 0010 0000	20
3º OUT: 0100 1110	4E

Codificando programa Assembly para o SAP-1 que realize a operação: [A5h + 2Ah - 06h].

End	Assembly	Cod. Máquina	Cod. Hex	Comentário
0h	LDA 7h	0000 0111	07	Carrega dado do endereço 7h
1h	OUT	1110 0000	E0	Envia para display
2h	ADD 8h	0001 1000	18	Soma dado do endereço 8h
3h	OUT	1110 0000	E0	Envia para display
4h	SUB 9h	0010 1001	29	Subtrai dado do endereço 9h
5h	OUT	1110 0000	E0	Envia para display
6h	HLT	1111 0000	F0	Termina o programa
7h	A5h	1010 0101	A5h	Dado
8h	2Ah	0010 1010	2Ah	Dado
9h	06h	0000 0110	06h	Dado
Ah	–	–	–	–
Bh	–	–	–	–
Ch	–	–	–	–
Dh	–	–	–	–
Eh	–	–	–	–
Fh	–	–	–	–



DISPLAY:

Resultado em Binário	Resultado em Hexadecimal
1100 1001	C9h