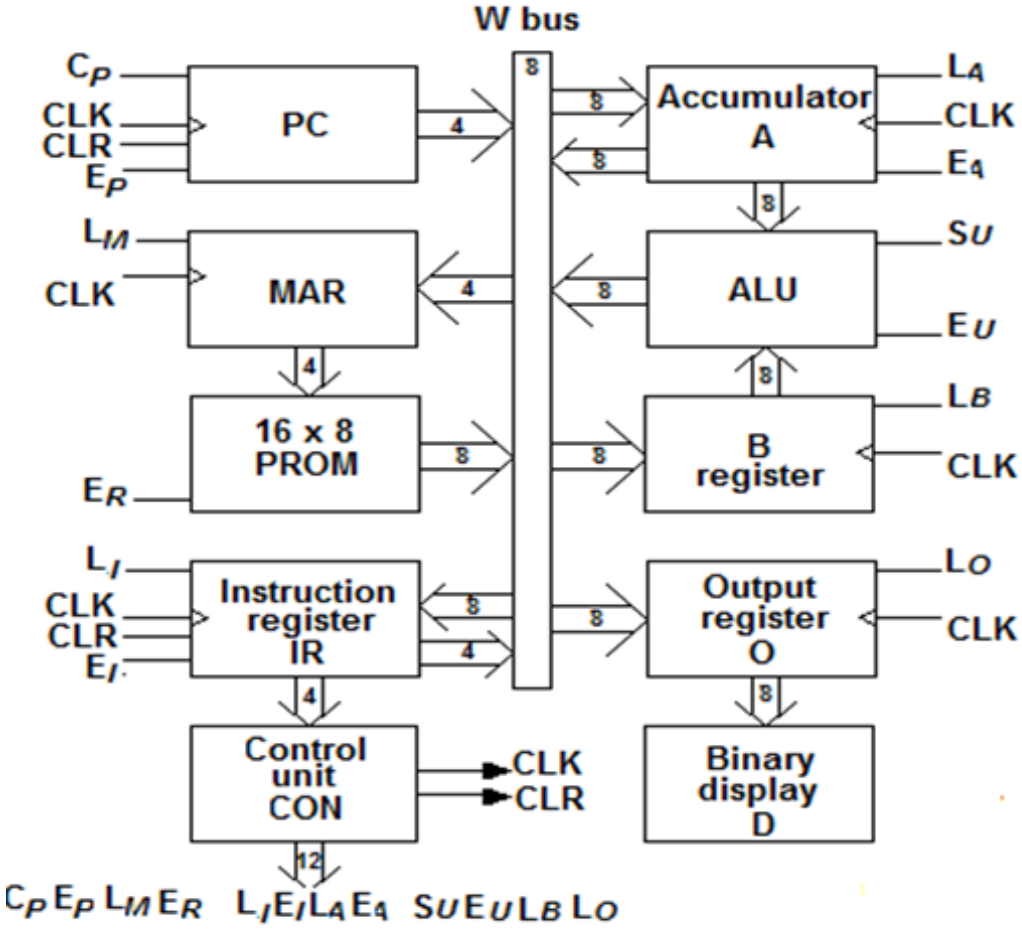
SAP-1

Arquitectura de un microcomputador según Albert Malvino

Por: Hugo Murillo

UNIVERSIDAD EAFIT
Abierta al mundo

Arquitectura SAP-1



Memoria de solo lectura

La 16x8 PROM. Almacena las 16 palabras de 8 bits cada una. Estas palabras, ubicadas en las direcciones 0 a F, se simbolizan por R_0, R_1, R_2..., R_f. Cuando E_R esta alta, la dirección de la palabra es leída en el bus W.

Registro de instrucciones

El contenido del registro de instrucciones se divide en dos palabras más pequeñas. Los cuatro bits más significativos (bits más significativos) son una salida de dos estados que va directamente a la unidad de control. Los cuatro bits menos significativos, corresponden a la dirección donde se encuentra el registro a trabajar de la memoria ROM

Unidad de Control

La unidad de control (CON) hace a un computador lo que es: una máquina automática para procesamiento de datos. Antes de que cada computadora funcione, CON envía una señal CLR para el registro de instrucciones y contador de programa. Esto elimina la última instrucción en el IR y reinicia el PC en 0000

CON también envía una señal de reloj a todos los registros, lo que sincroniza el funcionamiento del equipo, asegurando estados internos bien definidos. En otras palabras, todas las transferencias de registro y cambios se producen en el flanco positivo de una señal de reloj común.

Los bits de la parte inferior de CON forman una palabra de control de 12 bits:

$$CON = C_P E_P L_M E_R \; L_I E_I L_A E_A \; S_U E_U L_B L_O$$

Esta palabra determina como el registro va a reaccionar al próximo flanco positivo del reloj. Por ejemplo, un alto $\mathbf{C_P}$ significa que el contador de programa se incrementa, un alto $\mathbf{E_P}$ y $\mathbf{L_M}$ significa el contenido de la $\mathbf{P_C}$ se cargan en el MAR, etc.

CON es como el director de una sinfonía. Este sincroniza y orquesta todas las diferentes piezas de la computadora **SAP-1**. mediante la generación de diferentes rutinas, **CON** hace que los otros circuitos traigan una instrucción, carguen el acumulador con un número, añada dos números, etc.

Acumulador

El acumulador (A), es un grupo de flip-flops que almacena la respuesta mientras que el computador esta en funcionando. El acumulador tiene dos salidas. la salida de dos Estados va directamente a la unidad lógica - aritmética. La salida de tres estados se conecta al bus W. Por lo tanto, la palabra de 8 bits es llevada del acumulador a la ALU; la misma palabra aparece en el bús W cuando E_A es alta.

Unidad lógica aritmética

La unidad lógica aritmético (ALU) contiene sumador / restador de complemento a 2. Cuando la entrada $(\boldsymbol{S_U})$ es bajo, la ALU realiza la suma algebraica; de la siguiente manera:

ALU = A + B

Cuando $\boldsymbol{S_U}$ es alto, realiza la diferencia algebraica (complemento a 2):

ALU = A + B´

Registro B

Este registro se usa para hacer todas las operaciones con el acumulador.

Un alto L_B y un flanco positivo del reloj cargan la palabra del bus W en el registro B. La salida de dos estados del registro B va a la ALU, entregando el número que será sumado o restado del contenido del acumulador.

Registro de salida

Al final del proceso, el acumulador contiene el resultado del programa. En este punto necesitamos transferirla al mundo exterior. Acá es donde el registro de salida (**O**) es usado. Cuando E_A y L_O están en alto, la próxima señal positiva del reloj carga la palabra acumulador en el registro de salida.

Por lo general los computadores tienen gran cantidad de registros de salida, que están conectados por medios de *circuitos de interfaz a los* dispositivos periféricos. De esta forma la información procesada puede ser dirigida a impresoras, LCD, teclados, etc. (un circuito de interfaz por cada dispositivo).

Display binario

El display binario (**D**) es un dispositivo de 8 diodos emisores de luz (**LEDs**). Dado que cada LED se conecta con un flip-flop del registro de salida, el display binario muestra el contenido del registro de salida. Además después de haber transferido una respuesta del acumulador al registro de salida, podemos ver la respuesta en forma binaria.

Instrucciones

Un computador es un hardware inservible hasta que alguien lo programe. Esto quiere decir cargar paso por paso instrucciones en la memoria antes de iniciar un proceso.

LDA
ADD
SUB
OUT
HLT

LDA

LDA viene de "load the accumulator." Una completa instrucción LDA incluye una palabra ROM. $\boldsymbol{LDA\ R_8}$, Por ejemplo significa "load the accumulator with $\boldsymbol{R_8}$." Además, viene dado por:

$$\boldsymbol{R_B = 1111\ 0000}$$

La ejecución de $\boldsymbol{LDA\ R_8}$ resulta en :

$$\boldsymbol{A = 1111\ 0000}$$

Similarmente $\boldsymbol{LDA\ R_A}$ significa "load the accumulator with $\boldsymbol{R_A}$," $\boldsymbol{R_F}$ significa "load the accumulator with $\boldsymbol{R_F}$," y así.

ADD

ADD es otra instrucción SAP-1. Una completa instrucción ADD incluye una palabra ROM. Para la instrucción ADD $\boldsymbol{R_9}$ significa "ADD $\boldsymbol{R_9}$ al contenido del acumulador "; la suma en la ALU reemplaza el contenido original del acumulador.

Acá tenemos un ejemplo numérico, suponga que el decimal dos se encuentra en el acumulador y el decimal 3 en el registro $\boldsymbol{R_9}$, entonces:

$$\boldsymbol{A = 0000\ 00010}$$
$$\boldsymbol{R_9 = 0000\ 0010}$$

Durante la ejecución de $\boldsymbol{ADD\ R_9}$ sucede lo siguiente, primero $\boldsymbol{R_9}$ es cargado en el registro B para obtener:

$$\boldsymbol{B = 0000\ 0011}$$

Y casi instantáneamente el ALU forma la suma de A y B

$$\boldsymbol{ALU = 0000\ 0101}$$

ADD

Segundo, el contenido del ALU es cargado al acumulador para conseguir

$$\boldsymbol{A = 0000\ 0101}$$

Como puedes ver, el acumulador termina con la suma.

La anterior rutina es usada para todas las instrucciones ADD. La palabra de la ROM apuntada va al registro B y la salida de la ALU al acumulador. Este es el porque la ejecución de ADD $\boldsymbol{R_9}$ sustrae $\boldsymbol{R_9}$ del contenido del acumulador, la ejecución de ADD $\boldsymbol{R_F}$ sustrae $\boldsymbol{R_F}$, etcetera

SUB

SUB es otra instrucción SAP-1, y esta también esta incompleta sin una palabra ROM.

Por ejemplo , SUB RC significa “Sustraer RC de los contenidos del acumulador”; La diferencia formada en el ALU remplaza los contenidos originales del acumulador.

Para un ejemplo concreto, asuma que 7 decimales están en el acumulador y 3 decimales en el registro ROM RC. Entonces,

$$\boldsymbol{A = 0000\ 0111}$$
$$\boldsymbol{R_C = 0000\ 0011}$$

La ejecución del **SUB RC** se lleva a cabo de la siguiente manera. Primero, **RC** es cargada en el registro **B** para conseguir

$$\mathbf{B = 0000\ 0011}$$

Y casi instantáneamente el **ALU** forma la diferencia de A y B

$$\mathbf{ALU = 0000\ 0100}$$

SUB

Segundo, el contenido del **ALU** se carga en el acumulador para lograr

$$\mathbf{A = 0000\ 0100}$$

Entonces, el acumulador termina con la diferencia.

La rutina anterior aplica a todas las instrucciones **SUB** ; la palabra **ROM** abordado va al registro **B** y la salida de la **ALU** al acumulador. Esta es la razón por la ejecución de **SUB RC** sustrae **RC** a partir del contenido del acumulador, la ejecución de **SUB RE** extrae **RE** del acumulador, y así sucesivamente

OUT

La instrucción **OUT** indica al equipo de **SAP-1** para transferir los contenidos que se acumulan al registro de salida. Luego de que **OUT** a sido ejecutado, usted puede ver la respuesta del problema siendo solucionada.

OUT es completo por si mismo; por eso , usted no tiene que incluir una palabra ROM cuando esta usando **OUT** porque la instrucción no involucra la memoria.

HLT

HLT significa detener. Esta instrucción indica al computador que detenga el procesamiento de datos. **HLT** marca el fin de un programa similar a la forma en que un periodo marca el final de una frase. Usted debe usar una instrucción **HLT** al final de cada programa **SAP-1**; de otro modo, se obtiene la basura informática (respuesta sin sentido provocada por el procesamiento fuera de control).

HLT es completo por si mimo; por eso, usted no tiene que incluir una palabra **ROM** cuando esta usando **HLT** porque la instrucción no involucra la memoria.

Instrucciones

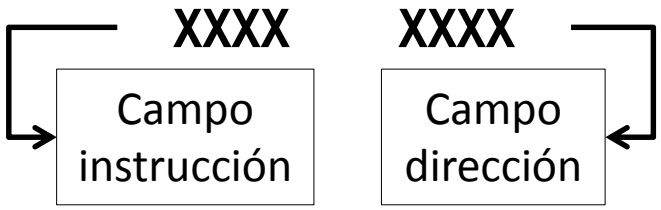
- ***Tabla 1. Conjunto de instrucciones.***

Nemotécnica	Operación
LDA	Carga el acumulador con la palabra ROM
ADD	Agrega palabra ROM al acumulador
SUB	Extrae palabra ROM del acumulador
OUT	Cargar registro de salida con la palabra acumulador
HLT	Detener

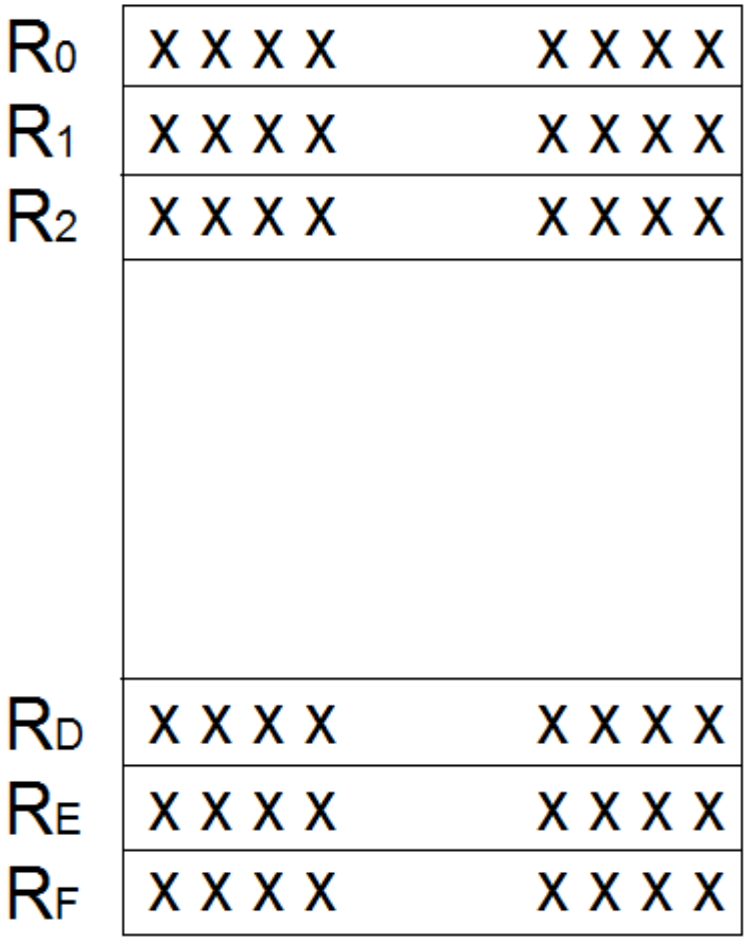
Instrucciones

- ***Tabla 2. Código de operación.***

Nemotécnica	Operación
LDA	0000
ADD	0001
SUB	0010
OUT	1110
HLT	1111



Contenido memoria ROM



Ejemplo 1

Éste es un programa SAP-1:

LDA R9

ADD RA

ADD RB

ADD RC

SUB RC

OUT

HLT

¿Que hace este Programa?

Solución Ejemplo 1

La primera instrucción carga el acumulador con R9

$$\boldsymbol{A = R_9}$$

La segunda instrucción suma RA al contenido del acumulador

$$\boldsymbol{A = R_9 + R_A}$$

Similarmente, las siguientes dos instrucciones suman RB y RC

$$\boldsymbol{A = R_9 + R_A + R_B + R_C}$$

La instrucción SUB resta RD del contenido del acumulador

$$\boldsymbol{A = R_9 + R_A + R_B + R_C - R_D}$$

La instrucción OUT carga esta palabra acumulador en el registro de salida, por lo tanto, la pantalla binaria muestra

$$\boldsymbol{D = A}$$

La instrucción HLT detiene el procesamiento de datos

Ejemplo 2

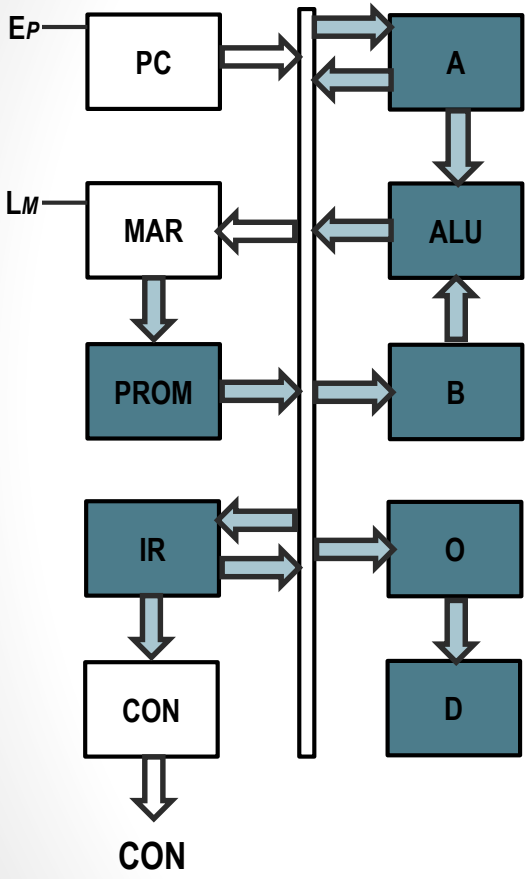
¿Cómo podría programar el SAP-1 para resolver el siguiente problema aritmético?

16+20+24-28+32

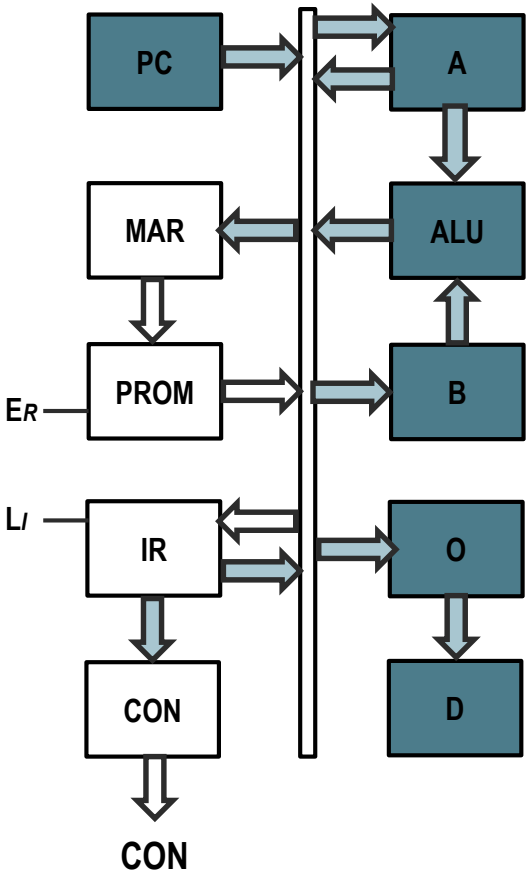
Solución Ejemplo 2

$R_0 = 0000\ 1001$	(LDA 9)
$R_1 = 0001\ 1010$	(ADD A)
$R_2 = 0001\ 1011$	(ADD B)
$R_3 = 0001\ 1100$	(ADD C)
$R_4 = 0010\ 1101$	(SUB D)
$R_5 = 1110\ XXXX$	(OUT)
$R_6 = 1111\ XXXX$	(HLT)
$R_9 = 0001\ 0000$	(16_{10})
$R_A = 0001\ 0100$	(20_{10})
$R_B = 0001\ 1000$	(24_{10})
$R_C = 0001\ 1100$	(28_{10})
$R_D = 0010\ 0000$	(32_{10})

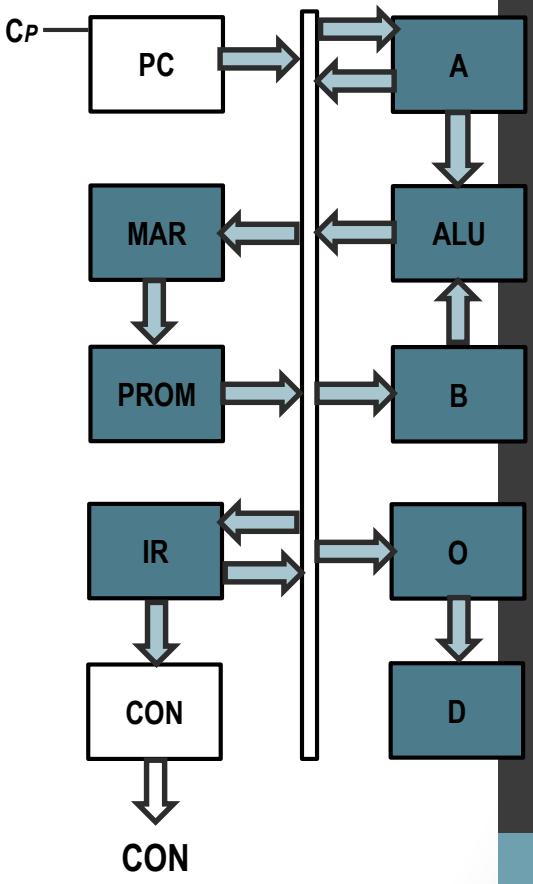
FETCH routine



a) Address phase

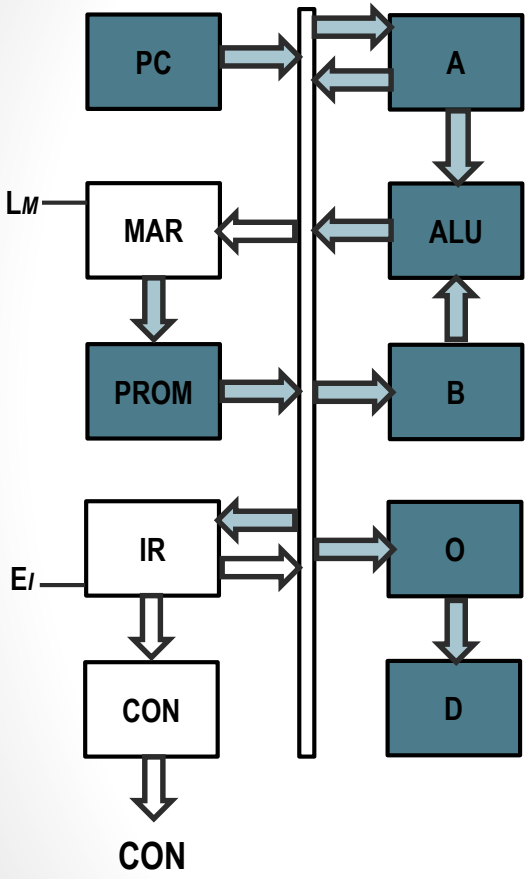


b) Memory phase

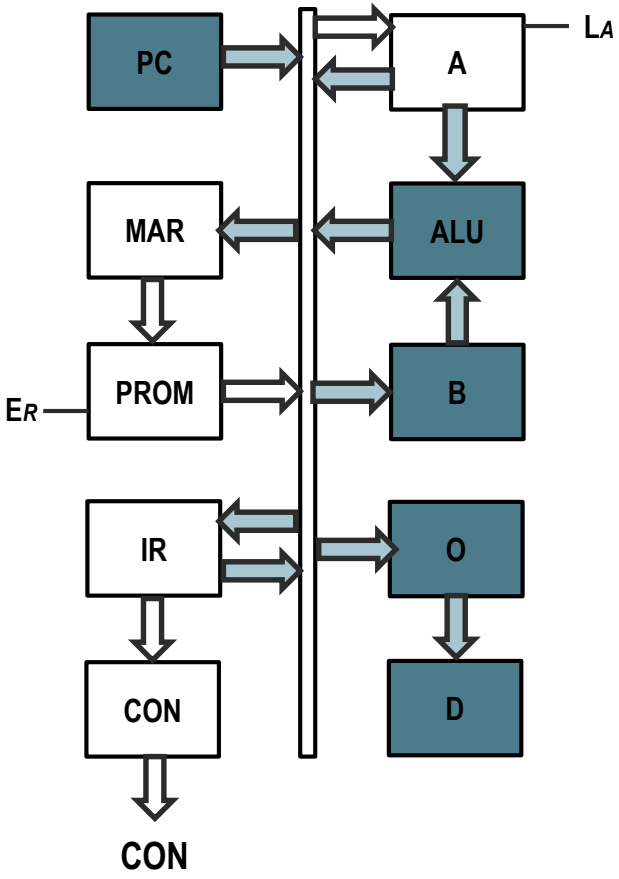


c) Increment phase

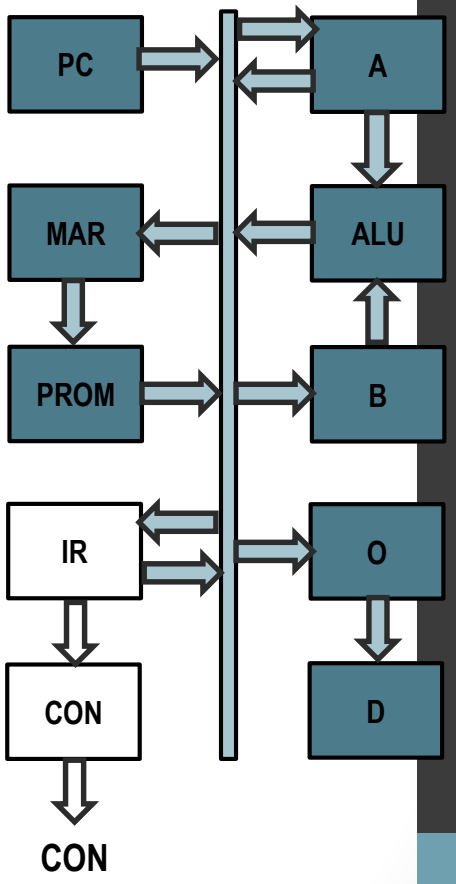
LDA routine



a) First execution phase

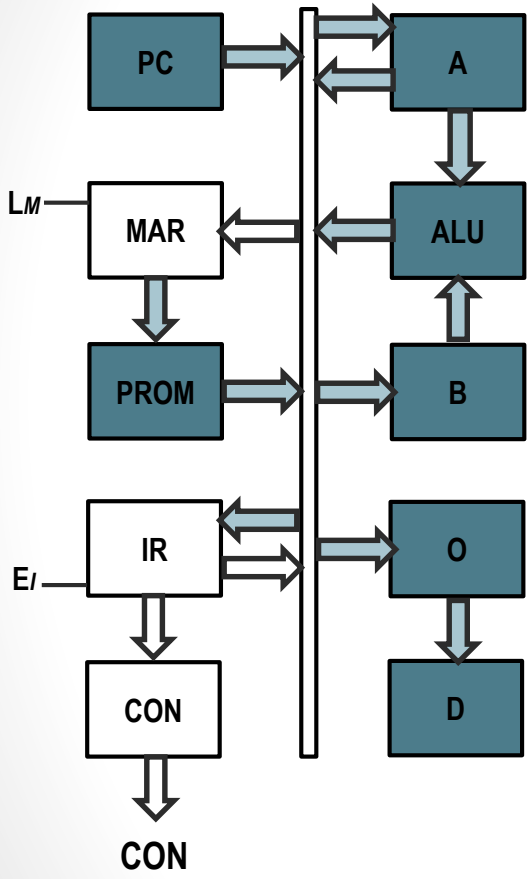


b) Second execution phase

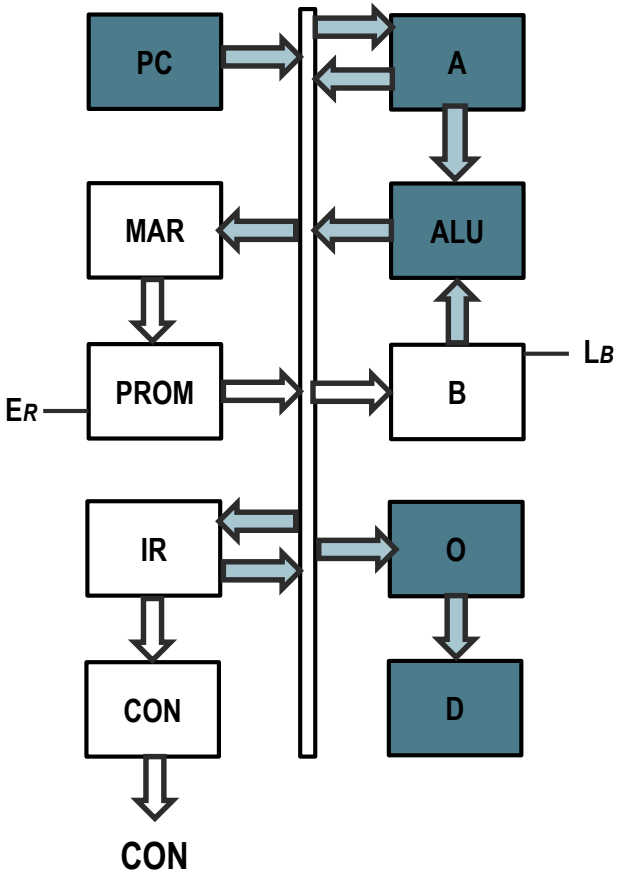


c) Final execution phase

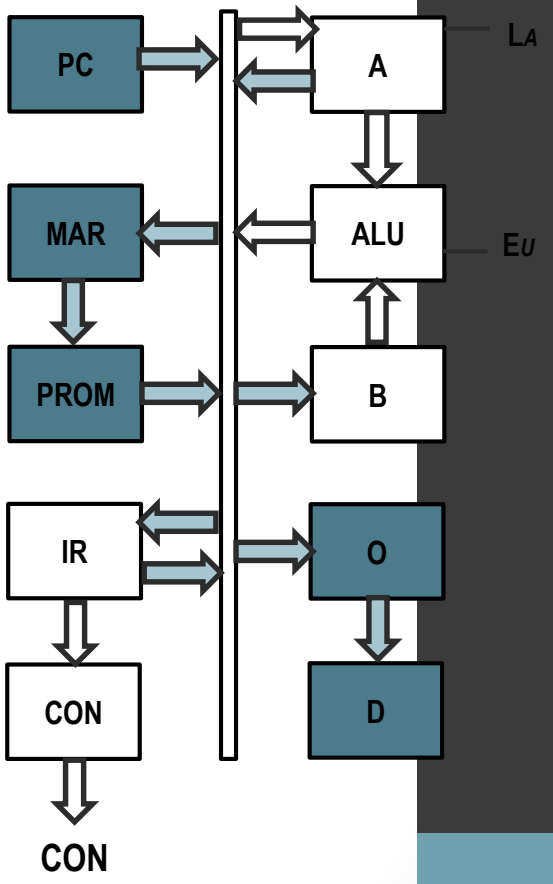
ADD and SUB routines



a) First execution phase

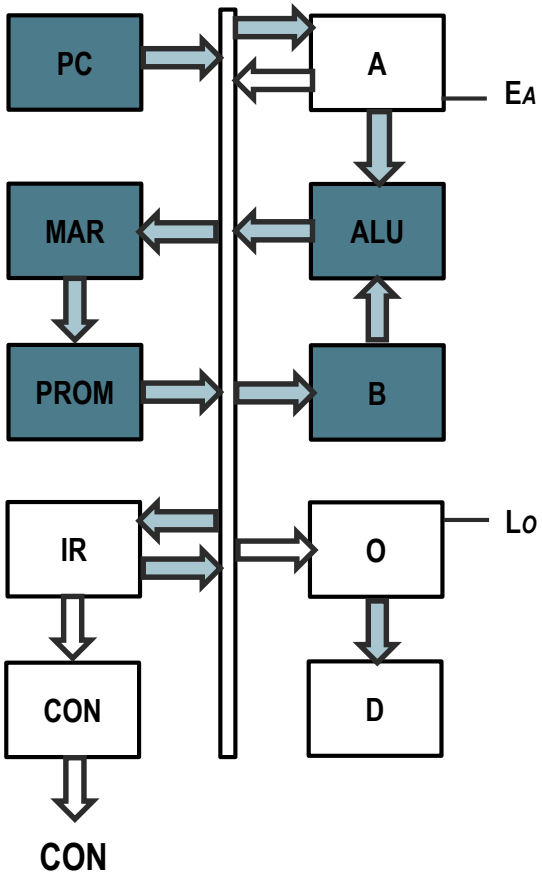


b) Second execution phase



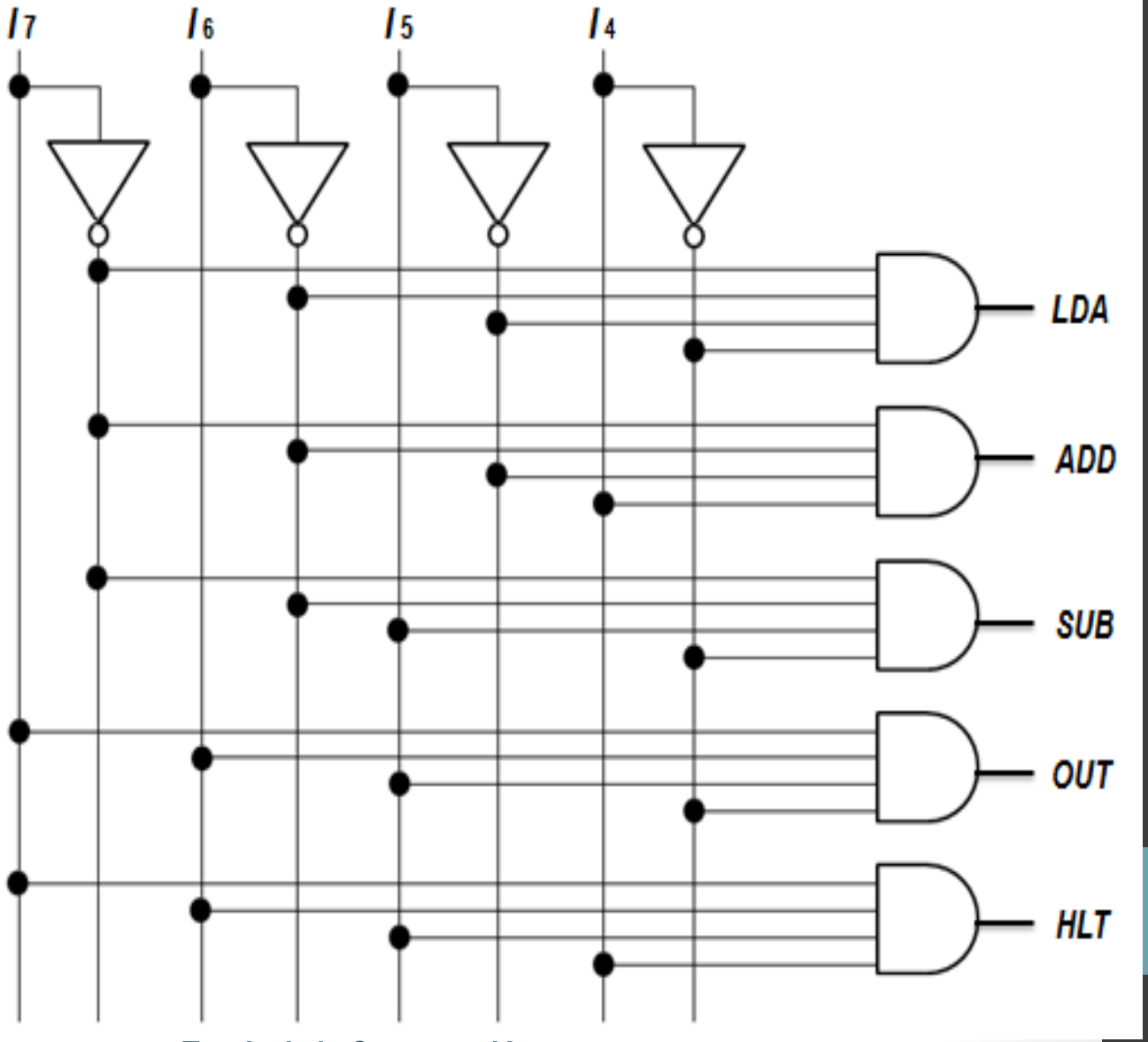
c) Final execution phase

First execution phase of OUT instruction



Instruction decoder

LDA	0000
ADD	0001
SUB	0010
OUT	1110
HLT	1111



Control matrix