6. HAFTA

BLM301

BİLGİSAYAR MİMARİSİ

Yrd. Doç. Dr. Salih GÖRGÜNOĞLU

sgorgunoglu@karabuk.edu.tr

KBUZEM
Karabük Üniversitesi
Uzaktan Eğitim Uygulama ve AraştırmaMerkezi

6. Temel Bilgisayar Tasarımı

Bir bilgisayar donanımı merkezi işlem birimi, RAM, giriş/çıkış arabirimi olmak üzere üç temel kısımdan oluşur. Merkezi işlem birimi (MİB) içerisinde, verileri işlemek için bir aritmetik ve mantık birimi, verileri saklamak için bir dizi kaydedici, ayrıca komutları getiren ve icra eden denetim devreleri bulunur. Bilgisayarın belleği komutları ve verileri saklamakta kullanılır. buna RAM adı verilir. Çünkü MİB, belleğin herhangi bir adresine herhangi bir anda erişebilir ve sonlu bir zaman içinde oradaki ikili bilgiyi alabilir. Giriş/çıkış arabirimi, dış ortamla bilgi alış-verişini denetleyen elektronik devreler içerir.

Gerçekleştirilen bu 16 bitlik bilgisayar tasarımıyla bilgisayar sistemlerinde donanım işlemlerini anlamak için gerekli temel bilgilerin öğrenilmesi, bir bilgisayar tasarımının nasıl yapılabileceğinin, bilgisayarın çalışma prensiplerinin öğrenilmesi amaçlanmıştır.

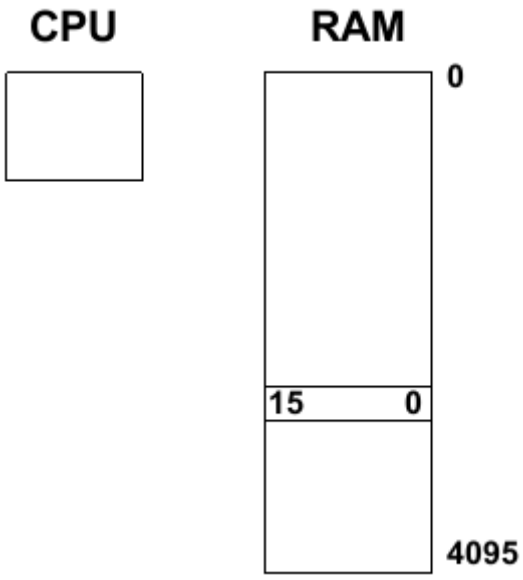
Temel Bilgisayar Organizasyonu ve Tasarımı

Bilgisayar tasarımının ilk adımını tasarlanacak bilgisayardan beklenenlerin, bilgisayarın ne tür yeteneklere sahip olacağının belirlenmesi oluşturmaktadır. Her faklı işlemci kendine has bir tasarımı vardır. Başka bir ifade ile çeşitli kaydedicileri, iletim yolları, mikroişlemleri, makine komutları vb vardır. Günümüz işlemcilerinin yapspı ise oldukça karmaşıktır. Yapılarında çok sayıda kaydedici, Tamsayı ve noktalı sayılarla işlem yapmak için birden fazla aritmetik işlem birimi, işlem hızını artırmak için aynı anda birden fazla komutu çalıştırma gibi (parallel işlem ve iş hattı yapıları) birimleri barındırmaktadır. Bununla birlikte bir işlemcinin nasıl çalıştığını anlama için basit bir işlemci modelini kullanmak daha mantıklı olacaktır. Burada Temel Bilgisayar (Basic Computer) modeli sunulacaktır.

Temel Bilgisayar

Temel Bilgisayarın iki önemli parçası vardır. Bunlar 4096 kelmelik bellek (RAM) birimi ve İşlemci (CPU) birimidir.

4096 = 2^{12}Belleği adreslemek için 12 bit yeterlidir. Dolayısı ile Adresleme ile ilgili olan PC, AR kaydedicileri 12 bitlik kaydediciler olacaktır. Belleğin her bir hücresi 16 bit(word) olacaktır.



Makine komutu (Machine instruction) işlemciye en yapması gerektiğni söyleyen ve ikili formatta yazılmış(10010110 gibi) ifadelerdir. Komutlar ardarda sıralanarak programlar oluşturulur. Başka bir ifade ile program makine komutlarının ardarda sıralanmasıdır.

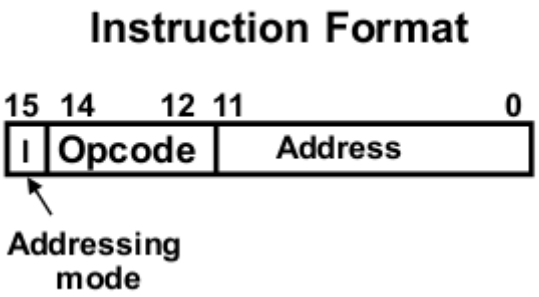
Programlar ve veriler aynı bellek üzerinde yer alırlar. CPU bellekten komut kodunu okur Komut kaydedicisine (IR) yükler. Kontrol birimi IR deki değere göre gerekli kontrol sinyallerini üreterek işlemin yapılmasını sağlar. Bir program içinde yer alan her bir makine komutunun çalışması üç adımda gerçekleştirilir. Bunlar

- Algetir(Fetch)
- Kodunu çöz(Decode)
- Çalıştır (Execute)

Komut Formatı (instruction Format)

Temel bilgisarda bir makine komutu 2 parçaya ayrılmıştır. Bu diğer işlemciler için farklı olabilir. Bu şekilde olmak zorunda değildir. İlk 12 bitlik kısım komutun kullandığı adresi tutar. Son 4 bitlik kısım opcode (operation code) işlem kodunu tutar. Ayrıca 15. Bit Bellek kullanımı (addressing mode) ile ilgili olup dogrudan(direct) ve dolaylı (indirect) olarak komutun belleği kullanacağını belitmektedir.

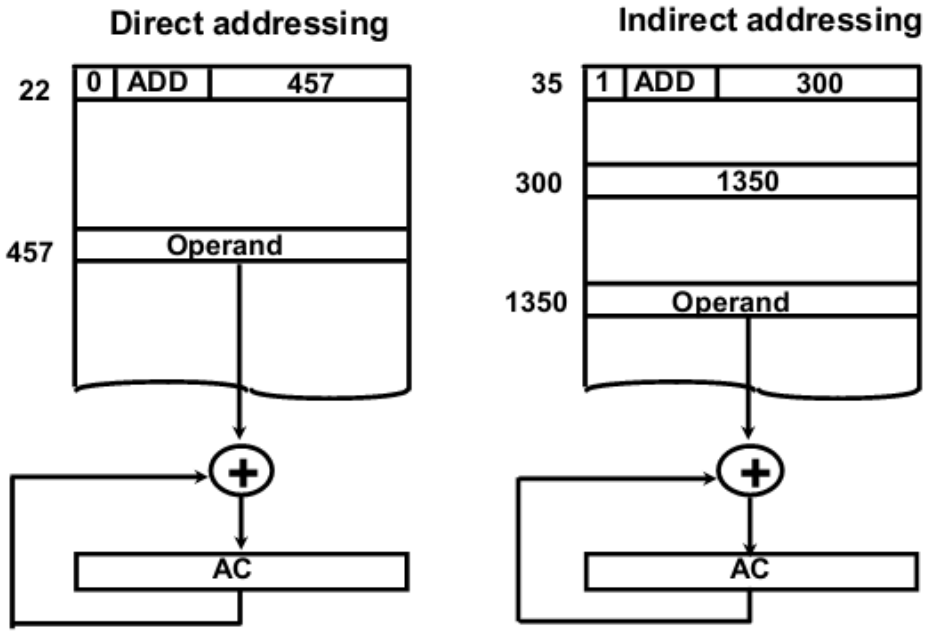
Temel bilgisayarın komut formatı şu şekildedir.



Örnek:

ADD 467 komutu hex olarak verilen doğrudan 467 adresindeki veri ile AC kaydedicisindeki veriyi toplamaktadır. (Soldaki şekil).

ADD 300 I komutu indirect olarak 300 adresindeki dolaylı olarak verilen ikinci adres 1350 adrsindeki değer ile AC kaydedicisini toplamaktadır.(sağdaki şekil)



Temel Bilgisayarda yer alan Kaydediciler ve görevleri

aşağıdaki donanım bileşenlerinden oluşmaktadır.

Temel bilgisayarda yer alan kaydediciler ve büyüklükleri şekil de görülmektedir. Şekil de de görüldüğü gibi Tasarlanan işlemci OUTR, INPR, PC, AR, AC, DR, TR, IR olmak üzere 8 kaydediciden oluşmaktadır.

Örnek program

Adress	Makine kodu	Komut(instruction)
100	2A15	LDA A15 ; A15 adresindeki veriyi AC kaydedicisine yükle
101	1A16	ADD A16 ; AC kaydedicindeki değerle A16 adresindeki değeri topla
103	4A17	STA A17 ; AC deki değeri A17 adresine yaz
104	7001	HLT ; İşlemciyi durdur

PC: Program sayıcı. İşlenecek olan komutun adresini tutar. Örneğin 100 adresinde LDA komutu var. PC 100 adresini tutar. Bu komutun çalışması bitince 101 adresini tutar. Bu şekilde devam eder. 12 bitliktir.

AR: adres kaydedicisidir. Belleği adresler. Yukarda verilen örnek programda AR kaydedicisi A15, A16, A17 adres değerlerini sırası ile tutacaktır. 12 bitliktir.

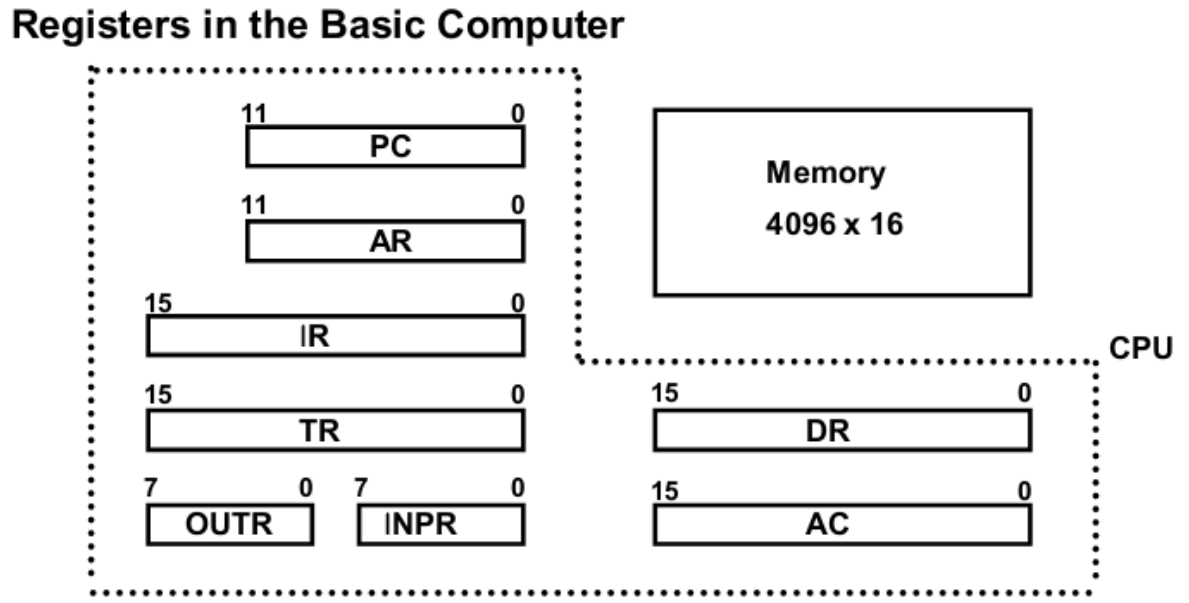
IR : Komut kaydedicisi Komut kodunu(makine kodu) tutar. Örneğin 2A15 makine kodu IR ye yüklenir. Bu LDA A15 anlamındadır. 16 bitliktir.

AC: Akumulator kaydedicisidir. Aritmetik ve lojik işlemlerde kullanılmaktadır. 16 bitliktir.

DR: Data kaydedicisidir. Yine AC ile beraber aritmetik ve lojik işlemlerde kullanılmaktadır. 16 bitliktir.

INPR: Giriş kaydedicisidir. Dış dunyadan değer almak için kullanılır. 8 bitliktir.

OUTR: Çıkış kaydedicisidir. Dış dünyaya değer göndermek için kullanılır. 8 bitliktir.

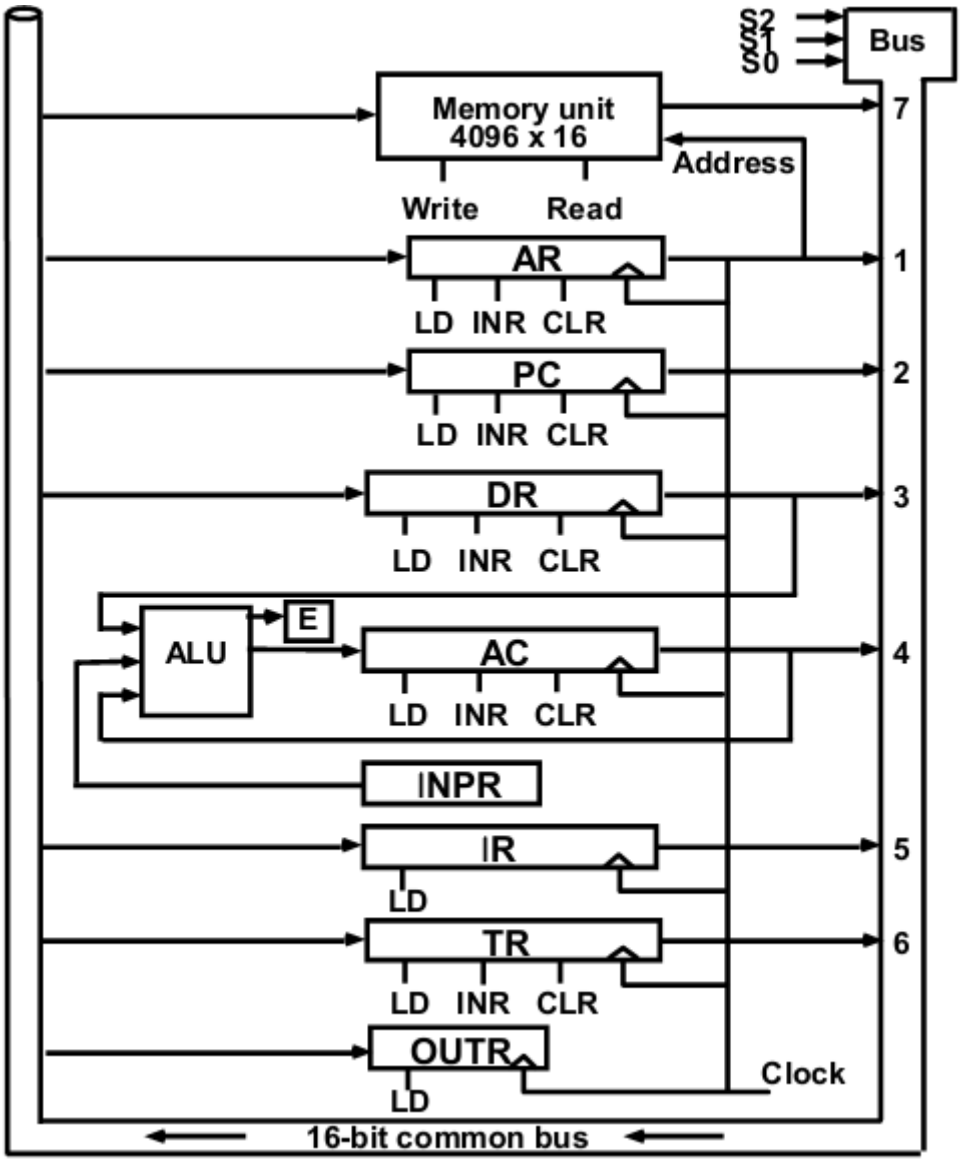


List of BC Registers

DR	16	Data Register	Holds memory operand
AR	12	Address Register	Holds address for memory
AC	16	Accumulator	Processor register
IR	16	Instruction Register	Holds instruction code
PC	12	Program Counter	Holds address of instruction
TR	16	Temporary Register	Holds temporary data
INPR	8	Input Register	Holds input character
OUTR	8	Output Register	Holds output character

Ortak Yol kullanımı

Bellek ve Kaydediciler ortak yol üzerinden biribirine bağlıdırlar. Kaydedicilerin ortak yola bağlanması şekil de gösterilmiştir.

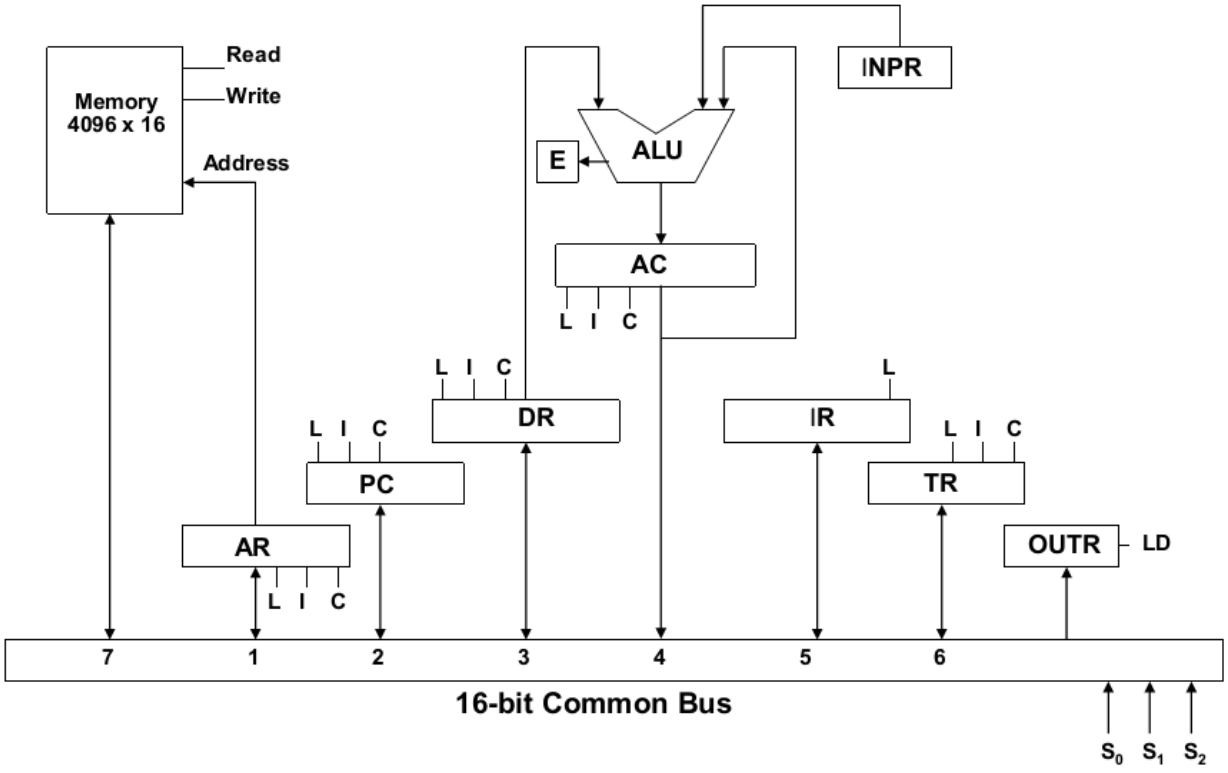


Her bir kaydedicinin Yükle(LD), Artır (INR), Temizle(CLR) kontrol girişleri yer almaktadır. IR ve OUTR kaydedicisinin sadece LD girişi vardır. Belleğin Read ve write kontrol girişleri bulunmaktadır. AR, PC, DR, IR, TR, ortak yoldan değer almakta ve ortak yola değer vermektedir. ALU (aritmetik mantık birimi) sadece AC, DR, ve INPR den değer almakta sonuç AC ye aktarılmaktadır. ALU ortak yoldan değer almadığna dikkat ediniz. Ayrıca kaydediciler ortak yolu gelişi güzel kullanmazlar. Yol kontrol birimi bu işlemi düzenler. Yol kontrol birimi şekilde görüldüğü gibi S2, S1, S0 girişlerine göre hangi kaydedicinin ortak yolu kullanacağını belirler. Girişlere S2=0,

S1=1, S0=1 uygulandığında ortak yolu DR kullanır. Diğer kaydediciler için de benzer düşünülebilir.

S_2	S_1	S_0	Register
0	0	0	x
0	0	1	AR
0	1	0	PC
0	1	1	DR
1	0	0	AC
1	0	1	IR
1	1	0	TR
1	1	1	Memory

Ortak yolun kullanımı ileri konularda daha ayrıntılı anlatılacaktır. Aşağıdaki şekilde yukarda verilen şekil basitleştirilmiştir. Ortak yol ile kaydediciler arasındaki iletişim çift yönlü gösyterilerek bu sağlanmıştır.

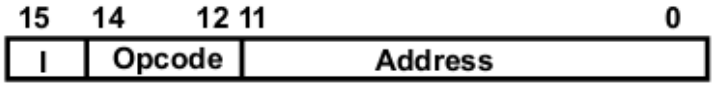


Temel Bilgisayarın Komut Kümesi

Temel bilgisatarda 3 çeşit komut formatı yer almaktadır.

Birincisi Bellek adresleyen(Memory reference instruction) komutlardır. Bu komutlar çalışırken bellekte bir yeri kullanırlar. Bu komurların formatı aşağıda gösterildiği gibidir.

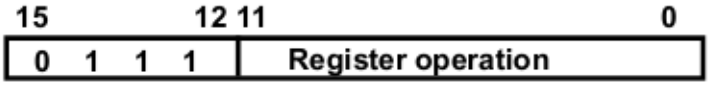
Memory-Reference Instructions (OP-code = 000 ~ 110)



Bu komutlarınopcode kısmı 0-6 aralığıda değer alır. Eğer I=1 ise yanı komutun kullandığı adres dolaylı (indirect) ise opcode I ile birlikte 8-E arası değer (hex olarak) alacaktır. İlk 12 bitlik kısım adresi ifade eder.

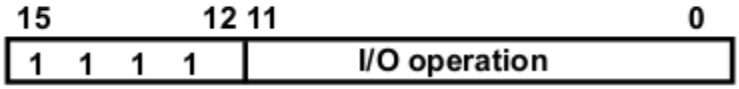
İkinci tip komutlar Kaydedici referanslı (bellek adreslemeyen -bellek kullanmayan - register reference instruction) komutlardır. Bu komutlar işlem yaparken belleğe ihtiyaç duymazlar. Bu yüzden Kaydedici referanslı komutlar(register reference instruction) olarak isimkendirilir Örnek olarak CLA komutu AC kaydedicisinin içeriğini temizler. Bu sırada bellek kullanılmaz. 12-15 arası bitler 7 değerini alır.

Register-Reference Instructions (OP-code = 111, I = 0)



Üçüncü tip komutlar Giriş- Çıkış (Input-Output Instruction) komutlarıdır. Bu komutlarda 12-15 bitler 1111 değerini (F) alırlar.

Input-Output Instructions (OP-code =111, I = 1)



Temel Bilgisayarın komut kümesi

Symbol	Hex Code I = 0	Hex Code I = 1	Description
AND	0xxx	8xxx	AND memory word to AC
ADD	1xxx	9xxx	Add memory word to AC
LDA	2xxx	Axxx	Load AC from memory
STA	3xxx	Bxxx	Store content of AC into memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and save return address
ISZ	6xxx	Exxx	Increment and skip if zero
CLA	7800		Clear AC
CLE	7400		Clear E
CMA	7200		Complement AC
CME	7100		Complement E
CIR	7080		Circulate right AC and E
CIL	7040		Circulate left AC and E
INC	7020		Increment AC
SPA	7010		Skip next instr. if AC is positive
SNA	7008		Skip next instr. if AC is negative
SZA	7004		Skip next instr. if AC is zero
SZE	7002		Skip next instr. if E is zero
HLT	7001		Halt computer
INP	F800		Input character to AC
OUT	F400		Output character from AC
SKI	F200		Skip on input flag
SKO	F100		Skip on output flag
ION	F080		Interrupt on
IOF	F040		Interrupt off

Komut kümesinin tamlığı(Instruction set completeness)

Bir bilgisayar sisteminde komut kümesinin tamlığından bahset mek için aşağıda verilen komut tiplernin olması gerekir.

Aritmetik, mantık ve kaydırma komutları (ör: ADD, INC, AND, CIR)

Veri transfer (veri aktarımı, taşıma, kopyalama) komutlar (Ör: LDA, STA)

Kontol motutları (ör: BUN, BSA, ISZ)

Giriş çıkış komutları. (INP, OUT)

Bu dört tip komut lar ile yapmak istediğimiz tüm işlemleri gerçekleştirebiliriz. Bu komut tiplerinden birinin olmaması komut kümesinde bir eksiklik oluşmasına neden olur. Ve yapmak istediğimiz bazı işlemleri gerçekleştiremeyiz.

Kontrol Birmi

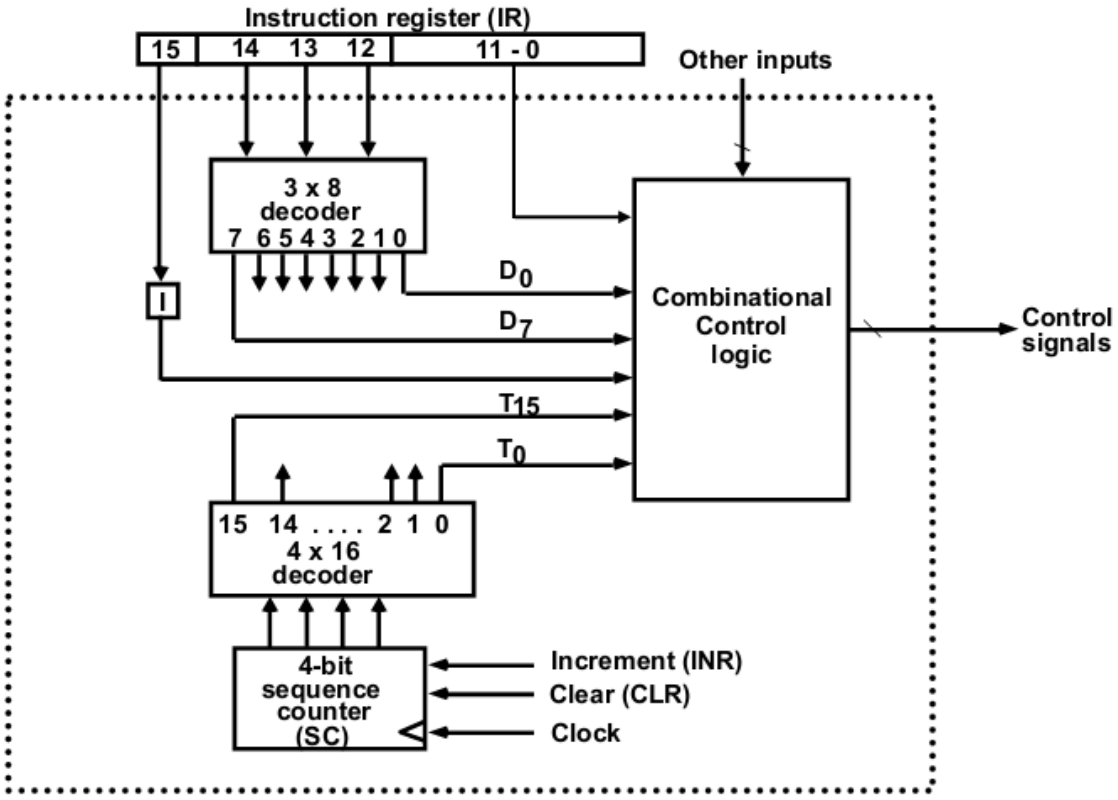
Kaydediciler üzerinde işlemler gerçekleştirilirken gerekli olan kontrol sinyallerinin üretilmesini sağlar. Bir bilgisayar sisteminde zamanlama bir clock sinyali ile gerçekleştirilirken, Kontrol birimi clock sinyaline göre ve gerekli olan kaydedici tipine göre kontrol sinyalini üretir. Her kaydedici için üretilen kontrol sinyali farklı olacaktır. Örneğin IR kaydedisinin LD girişine hangi şarlar altında ve hangi zamanda 1 verileceği kontrol birimi tarafında belirlenir. Aslında kontrol birimi bu işlemleri clock ile beraber gerçekleştirdiğinden zamanlama ve kontrol birimi şeklinde de isimlendirilebilir.
Kontrol birimi iki şekilde tasarlanır.

- Donanımsal (Harwired) olarak
- Mikroprogramlı (Microprogrammed) kontrol.

Ilkinde keydediciler için işlenen komuta göre üretilecek olan kontrol sinyalleri mantık devreleri ile üretilir. İkincisinde ise işlemci üzerinde bir kontrol belleği(micro code rom) yer alır. Bu bellekte gerekli kontrol sinyallerini üreten microprogram yer alır. Biz temel bilgisayar tasarımında ilk yöntemi göreceğiz.

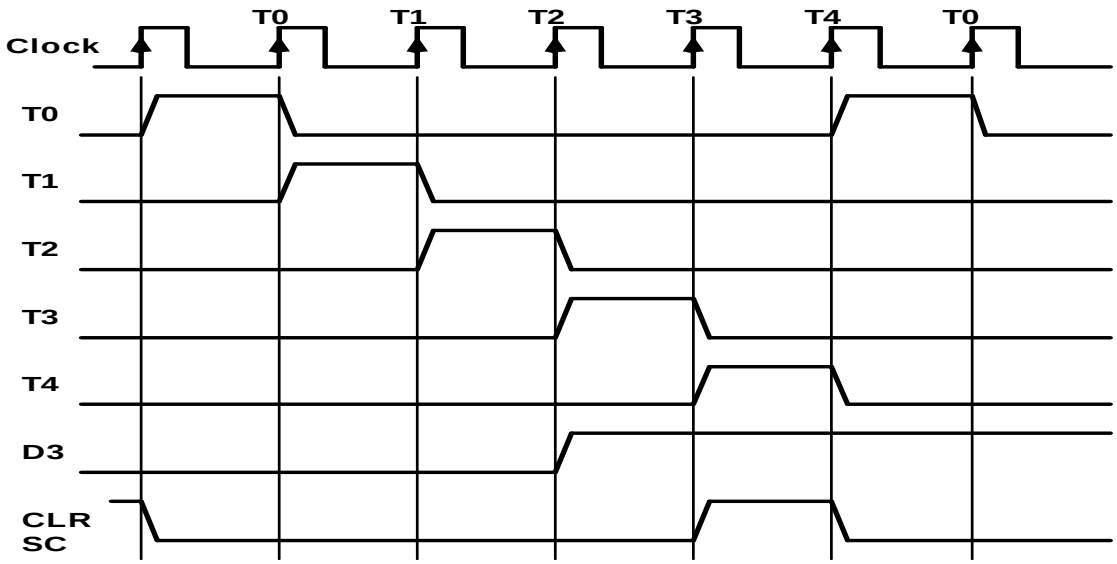
Temel bilgisayarın kontrol birimi aşağıdaki şekilde verilmiştir. Şekilden de görülebileceği gibi kontrol birimi kontrol sinyallerini üretirken IR Kaydedicisnin opcode kismı bir 3x8 decoder den geçrilmekte ve D0 … D7 arası sinylaller üretilmektedir. Aynı şekilde 4 bit sayıcı 4x16 decodere girmekte ve çıkışta T0 … T15 zamanlama sinyalleri üretilmektedir. Bu sinyallarde kontrol birimine girdi olmaktadır. I (indirect) biti de Kontrol birimine girdi olarak verilmiştir. Bunun yanında ihtiyaç

duyulan diğer girişler de olacaktır. Kontrol birimi bu girişlere göre gerekli kontrol sinyallerini üretmektedir.



4x16 decoder çıkışındaki elde edilen zamanlama (timing signal)aşağıda verilmiştir. SC sayıcısı 0-15 e kadar sayıyor. Buna göre T0…T15 sinyalleri üretiliyor. SC sayıcısı istendiğinde CLR girişinden sıfırlanabilir. Örneğin D3T4 üretidiğinde SC sıfırlanmak isteniyorsa aşağıdakigibi gösterilebilir.

D_3T_4: SC ← 0



Komut Çevrimi

Daha önce de söylendiği gibi bir komur üç aşamada çalıştırılır. Bunlar:

- Algetir(Fetch)
- Kodunu çöz(Decode)
- Çalıştır (Execute)

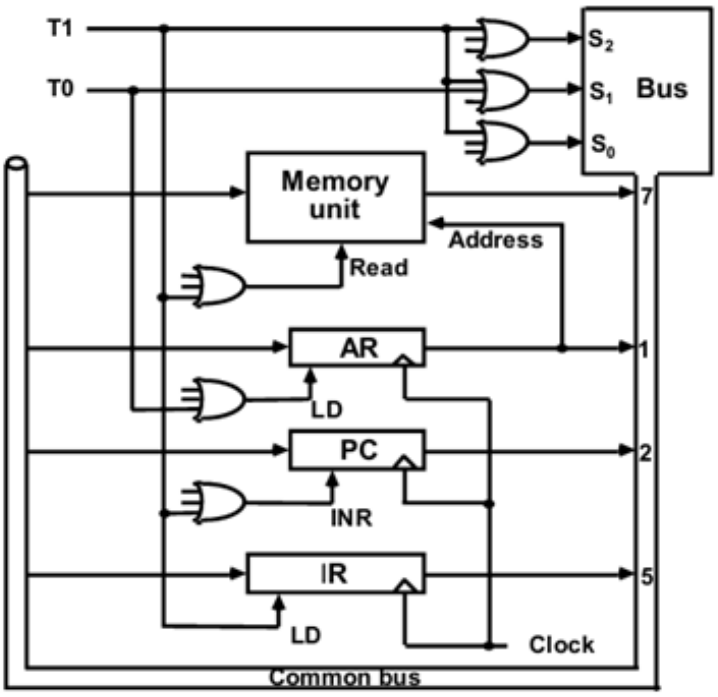
Programlar ve veriler aynı bellek üzerinde yer alırlar. CPU bellekten komut kodunu okur, komut kaydedicisine (IR) yükler. Buna Fetch denir. Daha sonra komutun kodu çözülür. Başka bir ifade ile Komutun ne olduğu belilenir. Bir sonrak aşama ise komutun çalıştırılmasıdır. Bu işlemler her bir komut için tekrarlanır.

Fetch ve Decode

Aşağıda verilen mikroişlemlerde T0 ve T1 zamanlarının başlaması ile fetch işlemi gerçekleştirilmekte, T2 zamanında ise decode işlemi yapılmaktadır. T0 anında PC deki adres değeri AR kaydedicisine alınmaktadır.PC çalışacak olan komutun yer aldığı bellek hücresinin adresini tuttuğu düşünüldüğünde, AR kaydedicisine PC değerinin aktarılması ile PC ile gösterilen adresteki komut koduna erişilmek istendiği anlaşılabilir. T1 anında AR kaydedicisinin gösterdiği bellek hücresindeki değer (Burada komut kodudur) IR kaydedicisine aktarılmaktadır. Ve daha sonra PC değeri

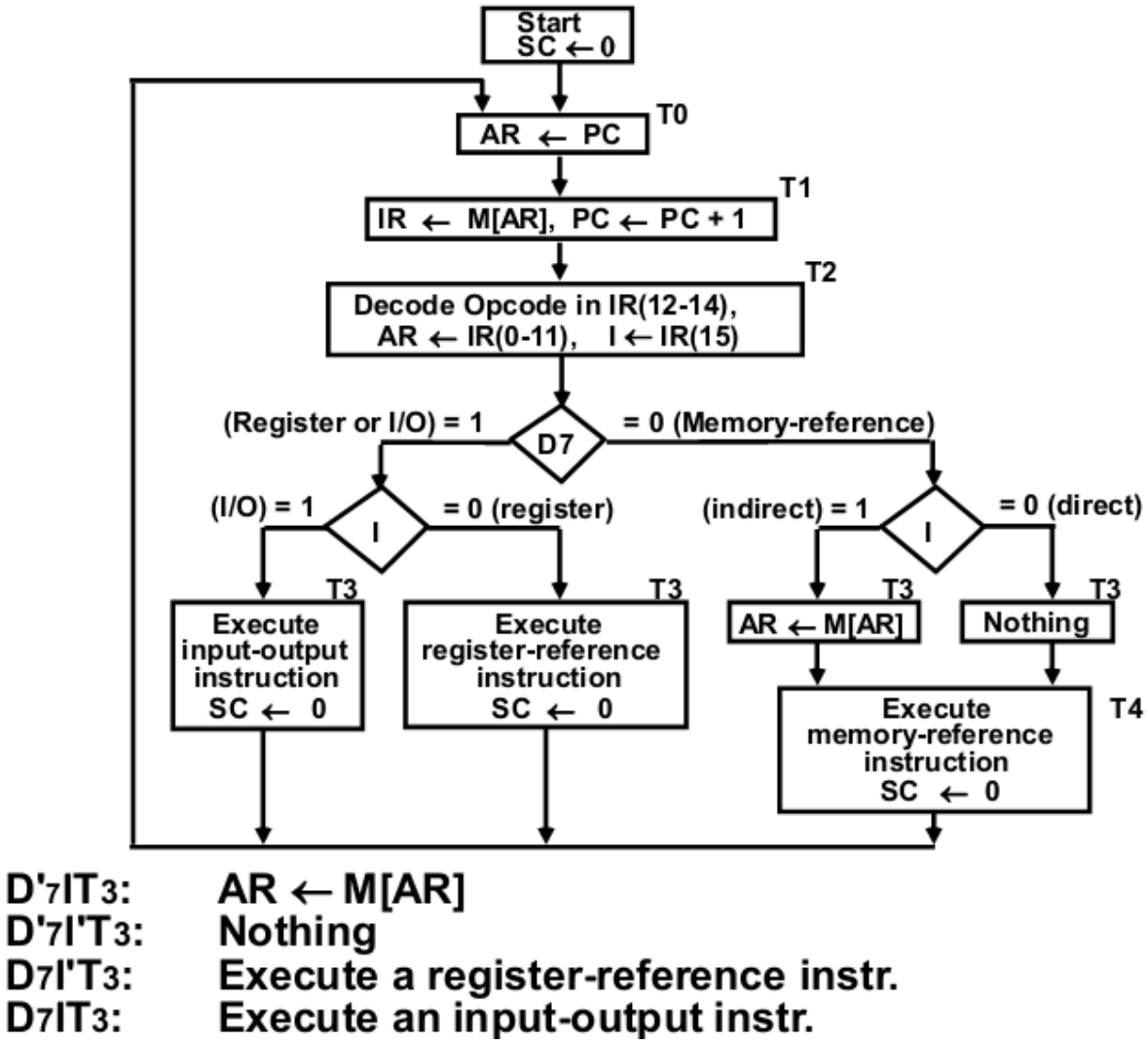
sonraki komutun bulunduğu adresi göstermesi için 1 artırılmaktadır. Bu ırada belleğin ortak yolu kullanması için, yol kontrol biriminin 7(111) çıkışını ürettiği görülmektedir. Çünkü belelekten okuma yapılmaktadır. Ve okunan değer ortak yol üzerinden IR ye aktarılmaktadır. Son olarak Decode(kodunu çöz) aşaması T2 de yapılmaktadır. T2 anında IR(12-14) bitleri decode edilerek D0-D8 arası kontrolsinyalleri üretilmektedir. D0-D6 arası üretildiğinde bunun bir bellek kullanan komut olduğu, D7 üretildiğinde kaydedici referanslı bir komut olduğu, D7 ile birlikte I=1 olması durumunda bunun bir giriş- çıkış komutu olduğu görülebilir. Baka bir ifade ile T2 anında komutun türü belirenmektedir. Komutun hangi komut olduğu belirlenmektedir.

T0: AR ← PC ($S_0S_1S_2$=010, T0=1)
T1: IR ← M [AR], PC ← PC + 1 (S0S1S2=111, T1=1)
T2: D0, . . . , D7 ← Decode IR(12-14), AR ← IR(0-11), I ← IR(15)



Aşağıda verilen akış şemasında komutun türünün nasıl belirlendiği görülmektedir. Başlangıcta sayıcı SC sıfırlanmaktadır. T= ve T! anlarında Fetch aşaması gerçekleşmakte, T2 anında ise Decode yapılmaktadır. Bu aşamadan sonr D7=0 ise bellek adresleyen bir komut, D7=1 ise kaydedici kullanan bir komut veya G/Ç komutudur. Eğer D7=0 ise T3 anında bellek adresleyen bir komut olduğundan I=0 ise bir şey yapmadan, I=1 ise indirect bellek adresleyeceği için AR kaydedicisine M[AR]

deki değer atılarak T4 anında işlem gerçekleştirilir ve yeni bir komut SC sıfırlanır. Eğer D7 =1 ise I=0 ise kaydedici referanslı komut T3 anında işletilir. Eğer I=1 ise G/Ç komutu T3 anında işletilir.



D'_7IT_3: $AR \leftarrow M[AR]$
$D'_7I'T_3$: Nothing
$D_7I'T_3$: Execute a register-reference instr.
D_7IT_3: Execute an input-output instr.

Kaydedici referanslı komutlar

Kaydedisi referanslı komutlar aşağıda veriilen bitler ile tanınırlar.

- D7=1, I=0 olmalıdır
- D7I'T3=1 ise kaydedici referanslı bir komuttur.
- IR kaydedicisinin ilk 12 biti (0..11) bitleri $B_i$ile gösterildiğinde Kaydedici referanslı komutlardan hangisi olduğu Bi ile belirlenir.

  Örneğin $D7I'T3B_0$=1 ise HLT komutudur.

  $D7I'T3B_{11}$=1 ise CLA komutudur.

Kaydedici referanslı komutlar ve komutlara ait mikroişlemler aşağıda verilmiştir.

$r = D_7 I'T_3$ => Register Reference Instruction
$B_i = IR(i)$, i=0,1,2,...,11

	r:	$SC \leftarrow 0$
CLA	rB_{11}:	$AC \leftarrow 0$
CLE	rB_{10}:	$E \leftarrow 0$
CMA	rB_9:	$AC \leftarrow AC'$
CME	rB_8:	$E \leftarrow E'$
CIR	rB_7:	$AC \leftarrow shr\ AC, AC(15) \leftarrow E, E \leftarrow AC(0)$
CIL	rB_6:	$AC \leftarrow shl\ AC, AC(0) \leftarrow E, E \leftarrow AC(15)$
INC	rB_5:	$AC \leftarrow AC + 1$
SPA	rB_4:	if (AC(15) = 0) then ($PC \leftarrow PC+1$)
SNA	rB_3:	if (AC(15) = 1) then ($PC \leftarrow PC+1$)
SZA	rB_2:	if (AC = 0) then ($PC \leftarrow PC+1$)
SZE	rB_1:	if (E = 0) then ($PC \leftarrow PC+1$)
HLT	rB_0:	$S \leftarrow 0$ (S is a start-stop flip-flop)

Bellek Adresleyen Komutlar (Memory reference İnstruction)

Bellek adresleyen Komutlar aşağıda verilmiştir.

Symbol	Operation Decoder	Symbolic Description
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR], E \leftarrow C_{out}$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC, PC \leftarrow AR + 1$
ISZ	D_6	$M[AR] \leftarrow M[AR] + 1$, if $M[AR] + 1 = 0$ then $PC \leftarrow PC+1$

Bu komutlar için tabloya bakıldığında dekoderden D0 üretilmiş ise AND, D1 üretidiğinde ADD, vb olarak D6 üretildiğinde ISZ çalıştırılmaktadır. Komutların Fetch ve Decode aşamaları ortaktır ve aşağıda beliritldiği üzere T0, T1,T2 zamanında gerçekleştirilmektedir. T3 zamanı ise indirect(dolaylı) adresleme için ayrılmıştır. Eğer

Dolaylı adresleme var ise I=1 dir ve T3 zamanıda dolaylı adres değeri bellekten alınarak AR kaydedicisine yerleştirilir.

Fetch ve Decode aşamaları tüm komutlar için ortaktır.

$T0: AR \leftarrow PC \quad (S_0S_1S_2=010, T0=1)$
$T1: IR \leftarrow M[AR], \quad PC \leftarrow PC + 1 \quad (S0S1S2=111, T1=1)$
$T2: D0, \ldots, D7 \leftarrow Decode\ IR(12\text{-}14), AR \leftarrow IR(0\text{-}11), I \leftarrow IR(15)$

İndirect aşaması

$$D'_7IT_3: \quad AR \leftarrow M[AR]$$

Bellek adresleyen komutların execute(çalıştır) aşamaları

AND to AC

D_0T_4:	$DR \leftarrow M[AR]$	Read operand
D_0T_5:	$AC \leftarrow AC \wedge DR, SC \leftarrow 0$	AND with AC

ADD to AC

D_1T_4:	$DR \leftarrow M[AR]$	Read operand
D_1T_5:	$AC \leftarrow AC + DR, E \leftarrow C_{out}, SC \leftarrow 0$	Add to AC and store carry in E

LDA: Load to AC

D_2T_4: $DR \leftarrow M[AR]$
D_2T_5: $AC \leftarrow DR, SC \leftarrow 0$

STA: Store AC

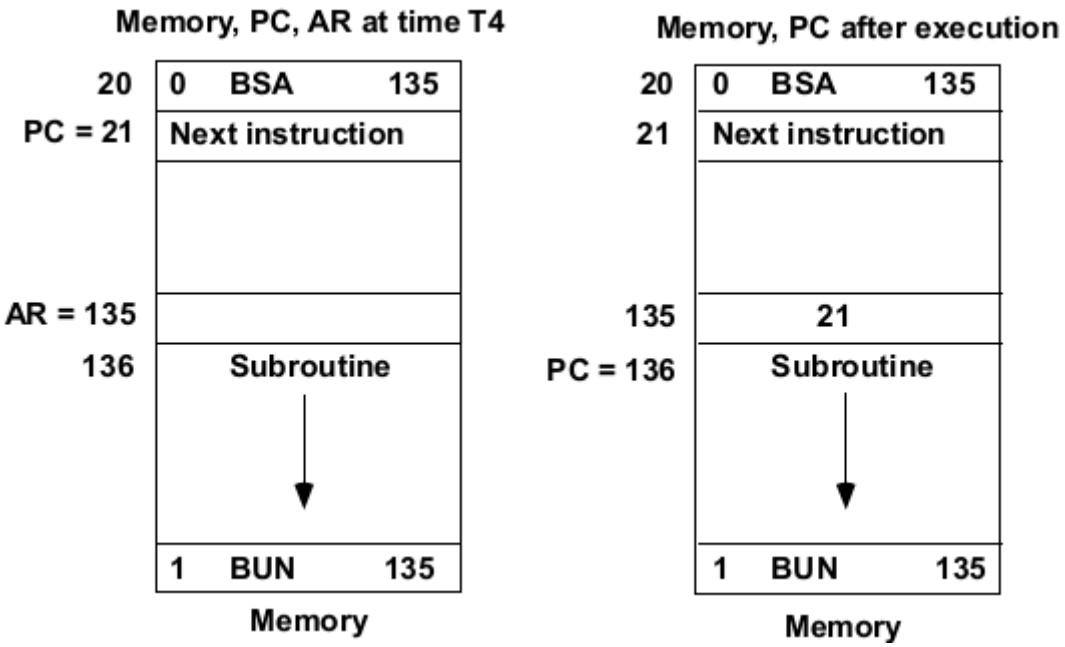
D_3T_4: $M[AR] \leftarrow AC, SC \leftarrow 0$

BUN: Branch Unconditionally

D_4T_4: $PC \leftarrow AR, SC \leftarrow 0$

BSA: Branch and Save Return Address

$M[AR] \leftarrow PC, PC \leftarrow AR + 1$



BSA:

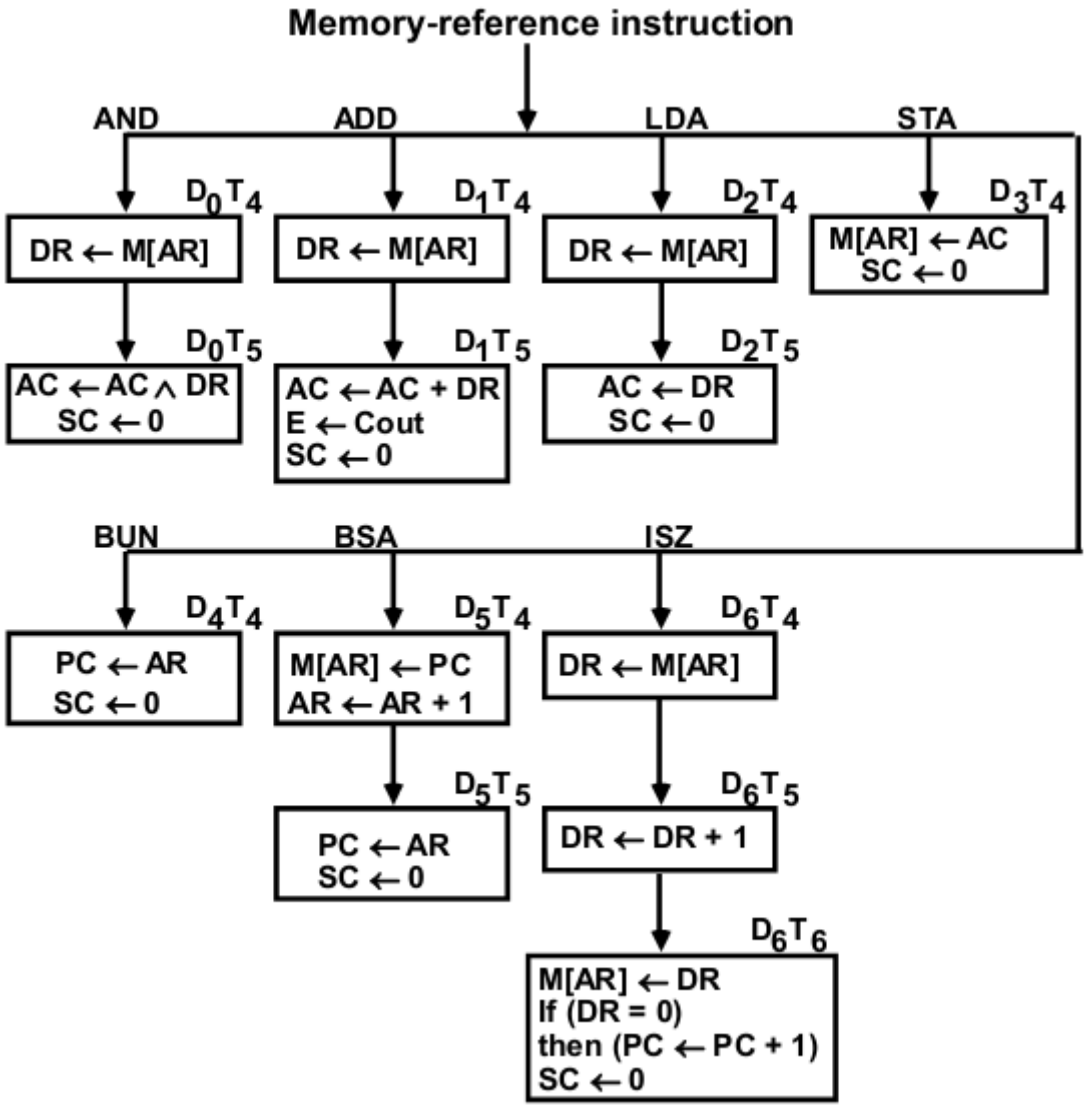
D_5T_4: $M[AR] \leftarrow PC,\ AR \leftarrow AR + 1$
D_5T_5: $PC \leftarrow AR,\ SC \leftarrow 0$

ISZ: Increment and Skip-if-Zero

D_6T_4: $DR \leftarrow M[AR]$
D_6T_5: $DR \leftarrow DR + 1$
D_6T_4: $M[AR] \leftarrow DR,\ \text{if } (DR = 0) \text{ then } (PC \leftarrow PC + 1),\ SC \leftarrow 0$

Bellek adresleyen komutlar (Memory reference instruction) için akış şeması

Görüleceği üzere bellek adresleyen komutların execute aşaması T4 zamanından başlamaktadır. İlgili decoder çıkışına göre hangi komutun çalışacağı belirlenmektedir.


Memory-reference instruction
AND
ADD
LDA
STA
D_0T_4
D_1T_4
D_2T_4
D_3T_4
DR ← M[AR]
DR ← M[AR]
DR ← M[AR]
M[AR] ← AC
SC ← 0
D_0T_5
D_1T_5
D_2T_5
AC ← AC ∧ DR
SC ← 0
AC ← AC + DR
E ← Cout
SC ← 0
AC ← DR
SC ← 0
BUN
BSA
ISZ
D_4T_4
D_5T_4
D_6T_4
PC ← AR
SC ← 0
M[AR] ← PC
AR ← AR + 1
DR ← M[AR]
D_5T_5
D_6T_5
PC ← AR
SC ← 0
DR ← DR + 1
D_6T_6
M[AR] ← DR
If (DR = 0)
then (PC ← PC + 1)
SC ← 0