SAP
(SIMPLE-AS-POSSIBLE)
COMPUTERS

Sadržaj

Uvod

SAP (Simple-As-Possible – Što jednostavnije moguće) računar je dizajniran za početnike. Glavna namena SAP-a je da predstavi sve najvažnije ideje vezane za računarske operacije bez prevelikog ulaženja u detalje. Ali čak i jednostavan računar kao SAP pokriva mnogo naprednih oblasti. Da vas ne bismo preopteretili sa previše informacija odjednom, izučavaćemo tri različite generacije SAP računara.

SAP–1 je prva faza u evoluciji ka modernom računaru. Iako primitivan, SAP-1 je veliki korak za početnika. Dobro izučite ovu oblast, savladajte SAP–1, njegovu arhitekturu, programiranje i elektronska kola. Onda ćete biti spremni za SAP–2.

---

10 – 1 Arhitektura

Slika 10-1 prikazuje *ahitekturu* (strukturu) SAP–1 računara organizovanog pomoću magistrala. Svi registrarski izlazi ka W magistrali mogu biti u tri stanja, ovo omogućava uređen prenos podataka. Svi ostali registarski izlazi mogu biti u dva stanja, ovi izlazi kontinualno upravljaju blokovima na koje su povezani.

Raspored blokova na slici 10-1 prikazuje registre koji se koriste u SAP–1. Iz ovog razloga nijednom nije pokušano da se sva upravljačka kola nalaze u jednom bloku zvanom upravljačka jedinica, sva ulazno – izlazna kola u drugom bloku koji se zove I/O jedinica itd.

Mnogi od registara sa slike 10-1 su poznati iz ranijih primera i diskusija. Ono što sledi je kratak opis svakog bloka, detaljnija objašnjenja dolaze kasnije.

Program Counter (Programski brojač)

Program se čuva na početku memorije gde je njegova prva instrukcija na binarnoj adresi 0000, druga na adresi 0001, treća na adresi 0010 itd. *Programski brojač*, koji je deo upravljačke jedinice, broji od 0000 do 1111. Njegov posao je da u memoriju upisuje adresu sledeće instrukcije koja će biti primljena i izvršena. To radi na sledeći način.

Programski brojač se resetuje na 0000 pre svakog pokretanja računara. Kada se računar pokrene, programski brojač upiše adresu 0000 u memoriju i zatim se uveća na 0001. Nakon što se prva instrukcija primi i izvrši programski brojač upiše adresu 0001 u memoriju i ponovo se uveća. Kada se druga instrukcija primi i izvrši programski brojač upiše adresu 0010 u memoriju. Na ovaj način programski brojač vodi računa o primanju i izvršavanju sledeće instrukcije.

Programski brojač je poput osobe koja prstom pokazuje na listu instrukcija i govori šta se izvršava prvo, drugo, treće itd. Zbog ovoga se programski brojač nekada zove i *pokazivač*, pokazuje na adresu u memoriji gde je smešteno nešto važno.

Input and MAR (Memory Address Register) (Ulaz i adresni registar memorije)

Ispod programskog brojača se nalazi *ulazni* i *MAR* blok. On obuhvata adrese i „data switch" registre (registri koji razmenjuju podatke) objašnjene u sekciji 9-4. Ovi switch registri, koji su deo ulazne jedinice, omogućavaju slanje četiri adresna bita i osam bita sa podacima u RAM. Kao što znate, instrukcije i „data words" (programske reči) se upisuju u RAM pre pokretanja računara.

MAR je deo SAP–1 memorije. Tokom pokretanja računara adresa u programskom brojaču prosledi se u MAR. Malo kasnije, MAR prosleđuje ovu četvorobitnu adresu u RAM gde se ona čita.

RAM memorija

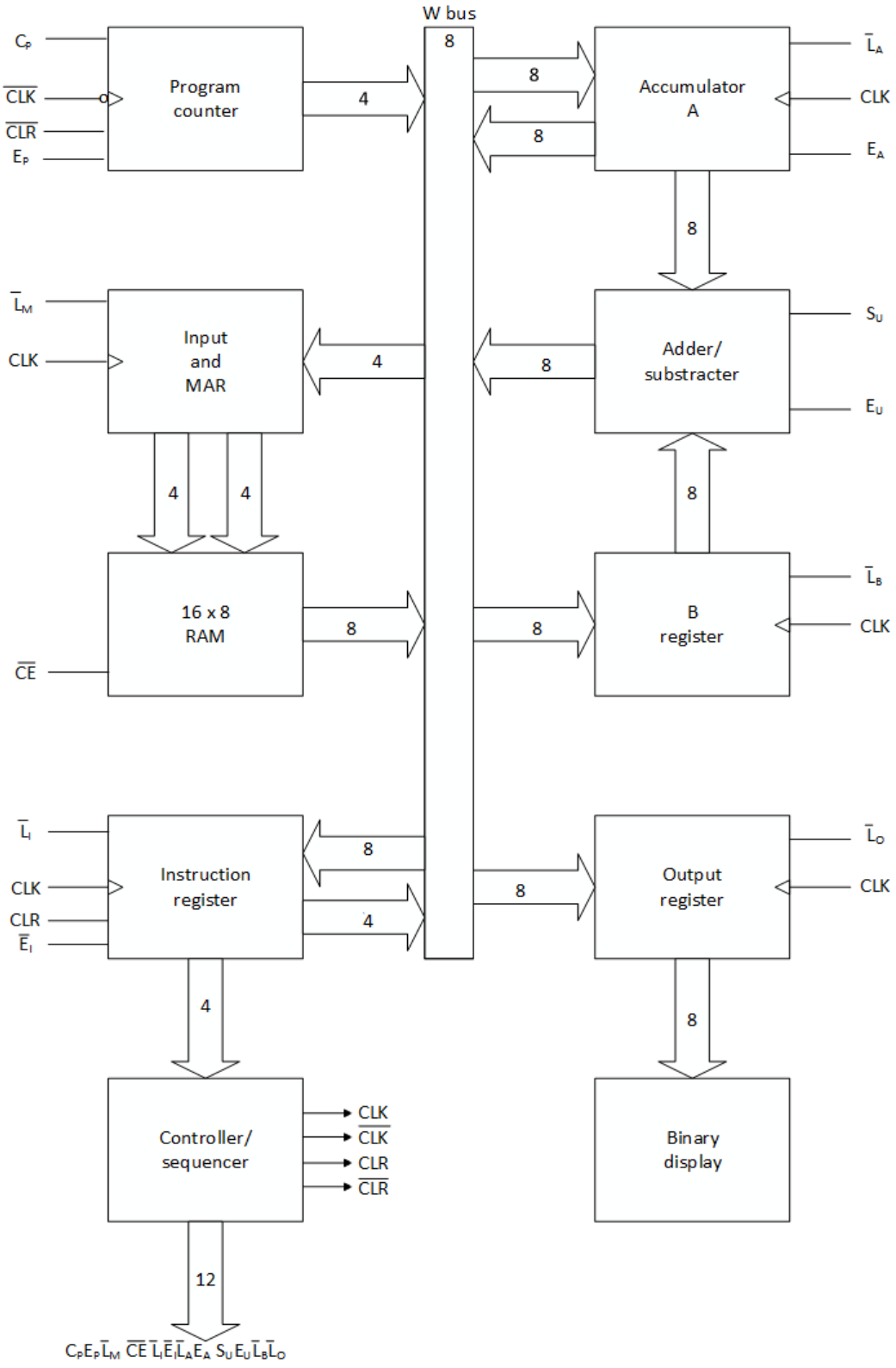
RAM je 16 × 8 statička TTL memorija. Kao što je objašnjeno u sekciji 9-4, možete programirati RAM pomoću adresa i data switch registara. Ovo omogućava da se sačuvaju program i podaci u memoriju pre pokretanja računara.

Tokom pokretanja računara RAM prima četvorobitnu adresu od MAR-a i čita je. Na ovaj način, instrukcija ili programska reč sačuvana u RAM-u je prosleđena na W magistralu za dalje korišćenje u nekom drugom delu računara.

Instrukcijski registar

Instrukcijski registar je deo upravljačke jedinice. Da bi prihvatio instrukciju iz memorije računar izvršava operaciju čitanja memorije. Ovo smešta sadržaj adresirane memorijske lokacije na W magistralu. U isto vreme, instrukcijski registar se sprema za čitanje na sledećoj gornoj ivici takta.

Sadržaj instrukcijskog registra je podeljen u dve polovine (engleski nibble – veličina od 4 bita). Gornja polovina je izlaz sa dva stanja koji je povezan direktno na blok nazvan „Controller – sequencer". Donja polovina je izlaz sa tri stanja koji je pročitan na W magistrali kad je to potrebno.



Slika 10–1 - SAP–1 arhitekture

Controller – Sequencer (Kontroler – Sekvencer)

Donji blok sa leve strane slike 10-1 sadrži *kontroler-sekvencer.* Pre svakog pokretanja računara $\overline{\text{CLR}}$ signal je poslat u programski brojač, a CRL signal u instrukcijski registar. Ovo resetuje programski brojač na 0000 i briše poslednju instrukciju iz instrukcijskog registra.

Taktni signal CLK je poslat u sve bafer (buffer) registre što sinhronizuje operacije računara obezbeđujući da se stvari dese u tačno određenom trenutku. Drugim rečima, sve razmene podataka između registara se dese na pozitivnoj ivici takta CLK. Obratite pažnju da i $\overline{\text{CLK}}$ takođe ide u programski brojač.

Dvanaest bita koji izlaze iz kontrolera-sekvencera formiraju programsku reč koja kontroliše rad ostatka računara (kao što supervizor govori drugima šta da rade). Dvanaest žica koje prenose upravljačku reč nazivaju se i *upravljačka magistrala*.

Upravljačka reč je u formatu

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

Ova reč određuje kako će registri reagovati na sledećoj pozitivnoj ivici takta. Na primer, visoko E_P i nisko $\bar{L}_M$ znače da će sadržaj programskog brojača biti upisan u MAR na sledećoj pozitivnoj ivici takta. Kao drugi primer, nisko $\overline{\text{CE}}$ i nisko $\bar{L}_A$ znače da će adresirana RAM reč biti prosleđena u akumulator na sledećoj pozitivnoj ivici takta. Kasnije ćemo objasniti vremenski dijagram da bismo videli kad i kako se dešavaju ovi transferi.

Akumulator

Akumulator (A) je bafer registar koji u sebi čuva međurezultate tokom rada računara. Na slici 10-1 akumulator ima dva izlaza. Izlaz sa dva stanja ide direktno na sabirač - oduzimač (adder - substracter). Izlaz sa tri stanja ide na W magistralu. Osmobitna reč iz akumulatora kontinualno upravlja sabiračem – oduzimačem, ista reč se pojavljuje na W magistrali kada se pojavi i visoko E_A.

The Adder – Substracter (Sabirač – Oduzimač)

SAP–1 koristi dvostruko komplementarni *sabirač – oduzimač*. Kada je S_U nisko na slici 10-1 suma iz sabirača – oduzimača je:

$$\mathbf{S = A + B}$$

Kada je S_U visoko na izlazu je razlika:

$$\mathbf{A = A + B'}$$

(Imajte u vidu da je dvostruki komplement ekvivalentan promeni decimalnog znaka.)

Sabirač – oduzimač je *asinhron* (radi bez takta), ovo znači da njegov sadržaj može da se promeni čim se promene i reči na ulazu. Kada je E_U visoko onda se taj sadržaj pojavi na W magistrali.

B registar

B registar je još jedan bafer registar. Koristi se kod aritmetičkih operacija. Nisko $\overline{L_B}$ i pozitivna ivica takta učitaju reč sa W magistrale u B register. Izlaz sa dva stanja iz B registra upravlja sabiračem – oduzimačem tako što prosleđuje broj koji treba da se sabere ili oduzme iz sadržaja akumulatora.

Output register (Izlazni registar)

Na primeru 8-1 objašnjen je izlazni registar. Na kraju rada računara akumulator sadrži odgovor na problem koji je bio rešavan. U ovom trenutku trebalo bi proslediti odgovor spoljašnjem svetu. Tad se koristi *izlazni registar*. Kada je E_A visoko i $\bar{L}_O$ nisko sledeća pozitivna ivica takta učitava reč iz akumulatora u izlazni registar.

Izlazni registar se često naziva i *izlazni port* jer obrađeni podaci „napuštaju“ računar kroz ovaj registar. U mikroračunarima izlazni portovi su povezani u *interfejs kola* (interface circuits) koja upravljaju perifernim uređajima poput štampača, monitora sa katodnom cevi i tako dalje. (Interfejs kola pripremaju podatke za upravljanje svakim uređajem).

Binary Display (Binarni displej)

Binarni displej je red od osam LED dioda. Zbog toga što je svaka dioda povezana na flip – flop na izlaznom portu binarni displej prikazuje sadržaj izlaznog porta. Nakon što smo izvršili transfer odgovora iz akumulatora na izlazni port možemo da vidimo odgovor u binarnom obliku.

Rezime

Upravljačka jedinica SAP–1 računara sadrži programski brojač, instrukcijski registar i kontroler – sekvencer koji proizvodi upravljačku reč, „čiste“ signale i signale takta. ALU (aritmetičko – logička jedinica) u SAP–1 sastoji se od akumulatora, sabirača – oduzimača i B registra. SAP–1 sadrži MAR i 16 × 8 RAM. Ulazno – izlazna jedinica (I/O) obuhvata ulazne programabilne prekidače (switch), izlazni port i binarni displej.

10 – 2 Instrukcijski set

Računar je beskorisna gomila hardvera sve dok ga neko ne isprogramira. Ovo podrazumeva učitavanje instrukcija u memoriju, korak po korak, pre nego što se računar pokrene. Pre nego što je moguće programirati računar potrebno je naučiti njegov set *instrukcija* odnosno osnovne operacije koje je moguće izvesti.

LDA

Kao što je opisano u sekciji 9, reči u memoriji se mogu označiti sa R_0, R_1, R_2 itd. Ovo znači da se R_0 čuva na adresi 0H, R_1 na adresi 1H, R_2 na adresi 2H i tako dalje.
LDA je oznaka za „load the accumulator“ (napuni akumulator). Potpuna LDA instrukcija obuhvatala heksadecimalnu adresa podatka koji će biti učitan. LDA 8H, na primer, znači „napuni akumulator sadržajem memorijske lokacije 8H“.

Dakle, ako imamo:

$$R_8 = 1111\ 0000$$

Izvršavanje LDA 8H bi rezultovalo sa:

$$A = 1111\ 0000$$

Slično tome, LDA AH znači „napuni akumulator sadržajem memorijske lokacije AH“, LDA FH znači „napuni akumulator sadržajem memorijske lokacije FH“ itd.

ADD

ADD je još jedna SAP–1instrukcija. Kompletna ADD instrukcija sadrži adresu reči koja će biti sabrana. Na primer, ADD 9H znači „dodaj sadržaj memorijske lokacije 9H sadržaju akumulatora, suma potom menja originalni sadržaj akumulatora.

Primer: Pretpostavimo da je decimalno 2 u akumulatoru i decimalno 3 je na memorijskoj lokaciji 9H. Zatim,

$$A = 0000\ 0010$$

$$R_9 = 0000\ 0011$$

Tokom izvršavanja instrukcije ADD 9H desiće se sledeće stvari. Prvo, R_9 se učitava u B registar da bi se dobilo:

$$B = 0000\ 0011$$

i skoro trenutno sabirač – oduzimač formira sumu A i B:

$$SUM = 0000\ 0101$$

Sledeće, suma se učitava u akumulator da bismo dobili:

$$A = 0000\ 0101$$

Gorenaveden postupak važi za sve ADD instrukcije, adresirana RAM reč šalje se u B registar, a izlaz sabirača – oduzimača u akumulator. Zbog ovoga izvršavanje instrukcije ADD 9H dodaje sadržaj R_9 sadržaju akumulatora, izvršavanje ADD FH dodaje sadržaj R_f u sadržaj akumulatora i tako dalje.

SUB

SUB je naredna SAP–1 instrukcija. Kompletna SUB instrukcija sadrži adresu reči koja će biti oduzeta. Na primer, SUB CH znači „oduzmi sadržaj memorijske lokacije CH od sadržaja akumulatora“, razlika iz sabirača – oduzimača zatim menja sadržaj akumulatora.

Na konkretnom primeru, pretpostavimo da je decimalan broj 7 u akumulatoru, a 3 u memorijskoj lokaciji CH. Onda,

$$A = 0000\ 0111$$

$$R_c = 0000\ 0011$$

Tokom izvršavanja instrukcije SUB CH desiće se sledeće stvari. Prvo, R_c se učitava u B registar da bi se dobilo:

$$B = 0000\ 0011$$

i skoro trenutno sabirač – oduzimač formira razliku A i B:

$$DIFF = 0000\ 0100$$

Sledeće, razlika se učitava u akumulator da bismo dobili:

$$A = 0000\ 0100$$

Gorenaveden postupak važi za sve SUB instrukcije, adresirana RAM reč šalje se u B registar, a izlaz sabirača – oduzimača u akumulator. Zbog ovoga izvršavanje instrukcije SUB CH oduzima sadržaj R_c od sadržaja akumulatora, izvršavanje SUB EH oduzima sadržaj R_e od sadržaja akumulatora i tako dalje.

OUT

Instrukcija OUT govori SAP -1 računaru da pošalje podatke iz akumulatora na izlazni port. Nakon što je instrukcija OUT izvršena možemo da vidimo odgovor na problem koji smo rešavali.

OUT je kompletna instrukcija sama po sebi. To znači da nije potrebno uključivati adresu kada se koristi OUT jer instrukcija ne piše nikakve podatke u memoriju.

HLT

HLT označa zaustavljanje (halt). Ova instrukcija govori računaru da zaustavi obradu podataka. HLT označava kraj programa, slično kao što tačka označava kraj rečenice. Obavezno je koristiti HLT na kraju svakog SAP–1programa, u suprotnom, dolazi do grešaka (dobijaju se besmisleni odgovori koji su uzrokovani beskonačnom obradom podataka).

HLT je kompletna instrukcija sama po sebi, nije potrebno uključivati RAM reči kada se koristi HLT jer ova funkcija ne upisuje ništa u memoriju.

Memorijsko – referencne instrukcije

LDA, ADD i SUB se zovu *memorijsko – referencne instrukcije* zato što koriste podatke koji su smešteni u memoriju. OUT i HLT, sa druge strane, nisu memorijsko – referencne instrukcije zbog toga što ne koriste podatke koji su smešteni u memoriju.

Mnemonics (Mnemonici)

LDA, ADD, SUB, OUT i HLT su setovi instrukcija za SAP–1. Skraćena imena instrukcija kao ovde navedena, nazivaju se *mnemonici*. Mnemonici su veoma popularni u računarstvu jer podsećaju na operaciju koja će se dogoditi kada se instrukcija izvrši. Tabela 10-1 sumira SAP–1 set instrukcija.

8080 i 8085

8080 je prvi široko korišćen mikroprocesor. Ima 72 instrukcije. 8085 je poboljšana verzija 8080 sa manje više istim instrukcijskim setom. Da bi SAP bio praktičan za upotrebu, njegove instrukcije su kompatibilne sa 8080/8085 instrukcijskim setom. Drugim rečima, SAP–1 instrukcije LDA, ADD, SUB, OUT i HLT su 8080/8085 instrukcije. Isto tako, SAP–2 i SAP–3 će biti deo 8080/8085 instrukcijskog seta. Učenjem SAP instrukcija bićete spremni za 8080 i 8085, dva veoma široko korišćena mikroprocesora.

Tabela 10-1 - SAP–1 instrukcijski set

Mnemonik	Operacija
LDA	Učitava RAM podatke u akumulator
ADD	Dodaje RAM podatke u akumulator
SUB	Oduzima RAM podatke od akumulatora
OUT	Šalje podatke iz akumulatora na izlazni port
HLT	Zaustavlja rad računara

Primer 10-1

SAP–1program u mnemonik formi:

Adresa	Mnemonik
0H	LDA 9H
1H	ADD AH
2H	ADD BH
3H	SUB CH
4H	OUT
5H	HLT

Podaci u „višoj“ memoriji su:

Adresa	Podatak
6H	FFH
7H	FFH
8H	FFH
9H	01H
AH	02H
BH	03H
CH	04H
DH	FFH
EH	FFH
FH	FFH

Šta svaka od ovih instrukcija radi?

Rešenje

Program je u „nižoj“ memoriji, lociran na adresi 0H do 5H. Prva instrukcija učitava sadržaj memorijske lokacije 9H u akumulator i samim tim sadržaj akumulatora postaje:

$$A = 01H$$

Druga instrukcija dodaje sadržaj memorijske lokacije AH sadržaju akumulatora da bismo dobili novu vrednost akumulatora:

$$A = 01H + 02H = 03H$$

Slično, treća instrukcija dodaje vrednost na memorijskoj lokaciji BH:

$$A = 03H + 03H = 06H$$

SUB instrukcija oduzima sadržaj memorijske lokacije CH:

$$A = 06H - 04H = 02H$$

Instrukcija OUT upisuje sadržaj akumulatora na izlazni port i binarni displej prikazuje:

$$0000\ 0010$$

Instrukcija HLT zaustavlja obradu.

10 – 3 Programiranje SAP–1

Da bi ste instrukcije i programske reči upisale u SAP–1memoriju moramo da koristimo neku vrstu koda kog računar može da razume. Tabela 10-2 prikazuje kod korišćen u SAP–1. Broj 0000 predstavlja LDA, 0001 ADD, 0010 SUB, 1110 OUT i 1111 HLT. Pošto ovaj kod govori računaru koju operaciju da izvrši on se naziva *operacioni kod* (op code).

Kao što je ranije objašnjeno, adresa i data switch-evi sa slike 9-7 omogućavaju programiranje SAP–1 memorije. Po dizajnu, ovi switch-evi proizvode 1 u gornjoj poziciji (U) i 0 u donjoj poziciji (D). Kada programiramo data switch-eve instrukcijom operacioni kod se šalje u gornju polovinu, a *operandi* (ostatak instrukcije) u donju polovinu.

Tabela 10 – 2 - SAP–1 operacioni kod

Mnemonik	Operacioni kod
LDA	0000
ADD	0001
SUB	0010
OUT	1110
HLT	1111

Na primer, pretpostavimo da želimo da učitamo sledeće instrukcije:

Adresa	Instrukcija
0H	LDA FH
1H	ADD EH
2H	HLT

Prvo, konvertuje se svaka instrukcija u binarni kod:

LDA FH = 0000 1111

ADD EH = 0001 1110

HLT = 1111 XXXX

U prvoj instrukciji 0000 je operacioni kod za LDA, a 1111 je binarni ekvivalent FH.
U drugoj instrukciji 0001 je operacioni kod za ADD, a 1110 je binarni ekvivalent za EH.
U trećoj instrukciji 1111 je operacioni kod za HLT, a XXXX su nebitni jer HLT nije memorijski referencna instrukcija.

Dalje, postave se adresa i data switch-evi:

Adresa	Podaci
DDDD	DDDD UUUU
DDDU	DDDU UUUD
DDUD	UUUU XXXX

Nakon što su postavljene sve adrese i programske reči pritisne se dugme za upis. Pošto D čuva binarnu 0, a U binarnu 1 prve tri memorijske lokacije sada imaju sledeće vrednosti.

Adresa	Vrednosti
0000	0000 1111
0001	0001 1110
0010	1111 XXXX

Asemblerski jezik radi sa mnemonicima kada upisuje program. *Mašinski jezik* radi sa stringovima nula i jedinica. Sledeći primeri pokazuju razliku između dva jezika.

Primer 10-2

Prevesti program iz primera 10-1 u SAP–1 mašinski jezik.

Rešenje

Program u primeru 10-1:

Adresa	Instrukcija
0H	LDA 9H
1H	ADD AH
2H	ADD BH
3H	SUB CH
4H	OUT
5H	HLT

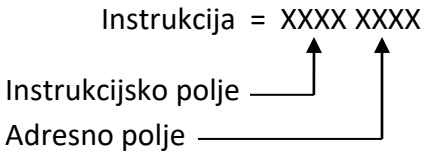
Program u asemblerskom jeziku je ovakav kao što se vidi. Da bi se pretvorio u mašinski jezik moramo ga prevesti u nule i jedinice na sledeći način:

Adresa	Instrukcija
0000	0000 1001
0001	0001 1010
0010	0001 1011
0011	0010 1100
0100	1110 XXXX
0101	1111 XXXX

Sada je program u mašinskom jeziku.

Bilo koji program kao gorenavedeni koji napisan u mašinskom jeziku se zove *objektni program*. Originalni program sa mnemonicima se zove *izvorni (source) program*. U SAP–1 operator prevodi izvorni program u objektni program kada programira adrese i data switch-eve.

Četiri MSB (Most Significant Bit) bita SAP–1 instrukcije mašinskog jezika određuju operaciju, a četiri LSB (Least Significant Bit) bita daju adresu. Ponekad MSB bite zovemo *instrukcijskim poljem*, a LSB bite *adresnim poljem*.



Primer 10-3

Kako isprogramirati SAP–1 da bi se rešio ovaj aritmetički problem?

16 + 20 + 24 -32

Brojevi su u decimalnom zapisu.

Rešenje

Jedan način je da se koristi program iz prethodnog primera, tako što će se podaci (16, 20, 24, 32) čuvati u memorijskim lokacijama od 9H do CH. Sa dodatkom 2, mogu se konvertovati decimalni podaci u heksadecimalne podatke da bi se dobila verzija za asemblerski jezik.

Adresa	**Sadržaj**
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

Verzija u mašinskom jeziku izgleda ovako:

Adresa	Sadržaj
0000	0000 1001
0001	0001 1010
0010	0001 1011
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

Obratite pažnju kako je program sačuvan pre podataka. Drugim rečima, program je u „nižoj“ memoriji, a podaci su u „višoj“. Ovo je izuzetno važno u SAP–1 zato što programski brojač pokazuje na adresu 0000 prilikom prve instrukcije, zatim na 0001 za drugu instrukciju i tako dalje.

Primer 10-4

Podeliti program i podatke iz prošlog primera u delove konverzijom u kratak heksadecimalan format.

Rešenje

Adresa	Sadržaj
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

Ova verzija programa i podataka se i dalje smatraju za mašinski jezik.

Ipak, negativni podaci se učitaju u dvostruko komplementarnom obliku. Na primer, -03H se unosi kao FDH.

---

10 – 4 Fetch cycle (Prihvatni ciklus)

Upravljačka jedinica je ključna za automatske operacije računara. Upravljačka jedinica generiše upravljačku reč koja prihvata i izvršava svaku instrukciju. Dok se svaka instrukcija prihvata i izvršava računar prolazi kroz različita *vremenska stanja*, T stanja (timing states, T states), odnosno periode tokom kojih se sadržaja registara menja.

Ring Counter (Kružni brojač)

Ranije smo diskutovali o SAP–1 kružnom brojaču (šematski dijagram na slici 8-16). Slika 10-2a simbolizuje kružni brojač koji ima izlaz:

$$T = T_6 T_5 T_4 T_3 T_2 T_1$$

Na početku rada računara kružna reč je:

$$T = 000001$$

Naredni taktni signali će menjati kružnu reč na sledeći način:

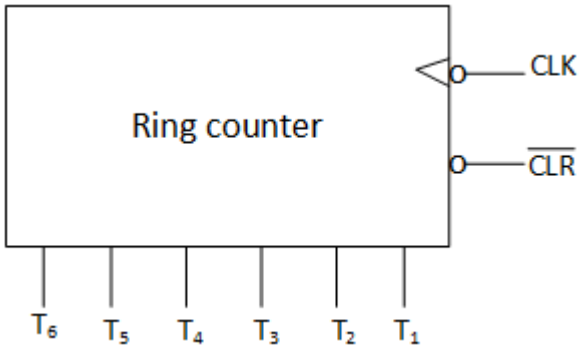
$$T = 000010$$
$$T = 000100$$
$$T = 001000$$
$$T = 010000$$
$$T = 100000$$

Zatim se kružni brojač resetuje na 000001 i ciklus se ponavlja. Svaka kružna reč predstavlja jedno T stanje.

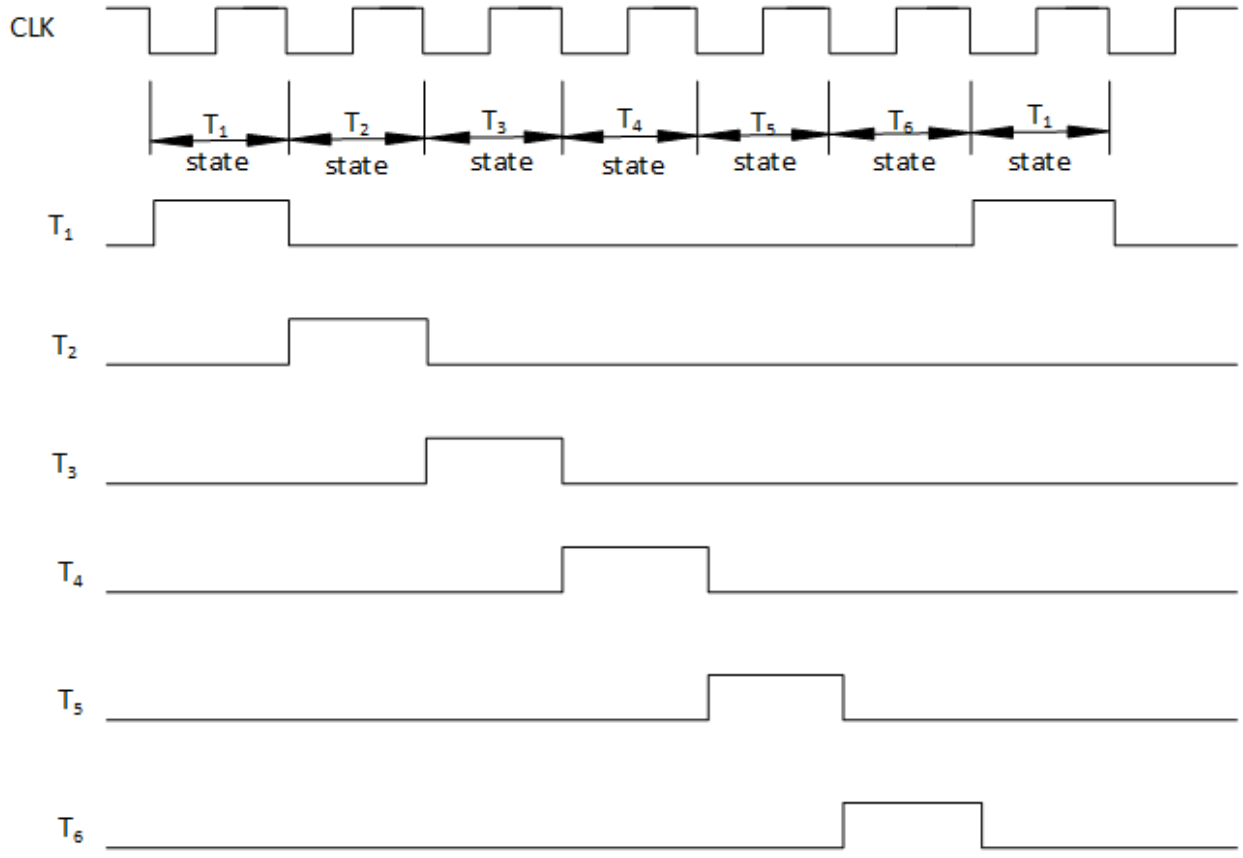
Slika 10-2b prikazuje vremenski dijagram kružnog brojača. Početno stanje T_1 počinje sa negativnom ivicom takta a završava se sa sledećom negativnom ivicom takta. Tokom ovog T stanja T_1 bit u kružnom brojaču je na logičkoj jedinici.

Tokom sledećeg stanja, T_2 je na jedinici zatim T_3 i tako dalje. Kao što se može videti, kružni brojač ima šest T stanja. Svaka instrukcija se prihvata i izvršava tokom ovih šest stanja.

Obratite pažnju da se pozitivna ivica CLK takta desi na polovini T stanja. Značaj ovoga će biti diskutovan kasnije.



Slika 10-2a - Kružni brojač: simbol



Slika 10-2b - Kružni brojač: takt i vremenski signal

Adresno stanje

T_1 stanje se naziva *adresno stanje* zato što se adresa u programskom brojaču (PC) šalje u memorijsko adresni registar (MAR) tokom ovog stanja. Slika 10-3a prikazuje sekcije računara koje su aktivne tokom ovog stanja (aktivni delovi su svetli, neaktivni su tamni).

Tokom adresnog stanja E_P i $\bar{L}_M$ su aktivni, svi ostali upravljački biti su neaktivni. Ovo označava da kontroler – sekvencer šalje upravljačku reč:

$$\boldsymbol{CON} = C_P E_P \bar{L}_M \overline{CE} \;\; \bar{L}_I \bar{E}_I \bar{L}_A E_A \;\; S_U E_U \bar{L}_B \bar{L}_O$$
$$= 0 \;\; 1 \;\; 0 \;\; 1 \;\;\; 1 \;\; 1 \;\; 1 \;\; 0 \;\;\; 0 \;\; 0 \;\; 1 \;\; 1$$

tokom ovog stanja.

Inkrementalno stanje

Slika 10-3b prikazuje aktivne delove SAP–1 tokom T_2 stanja. Ovo se zove *inkrementalno stanje* jer se programski brojač uvećava za jedan. Tokom ovog stanja kontroler – sekvencer šalje upravljačku reč:

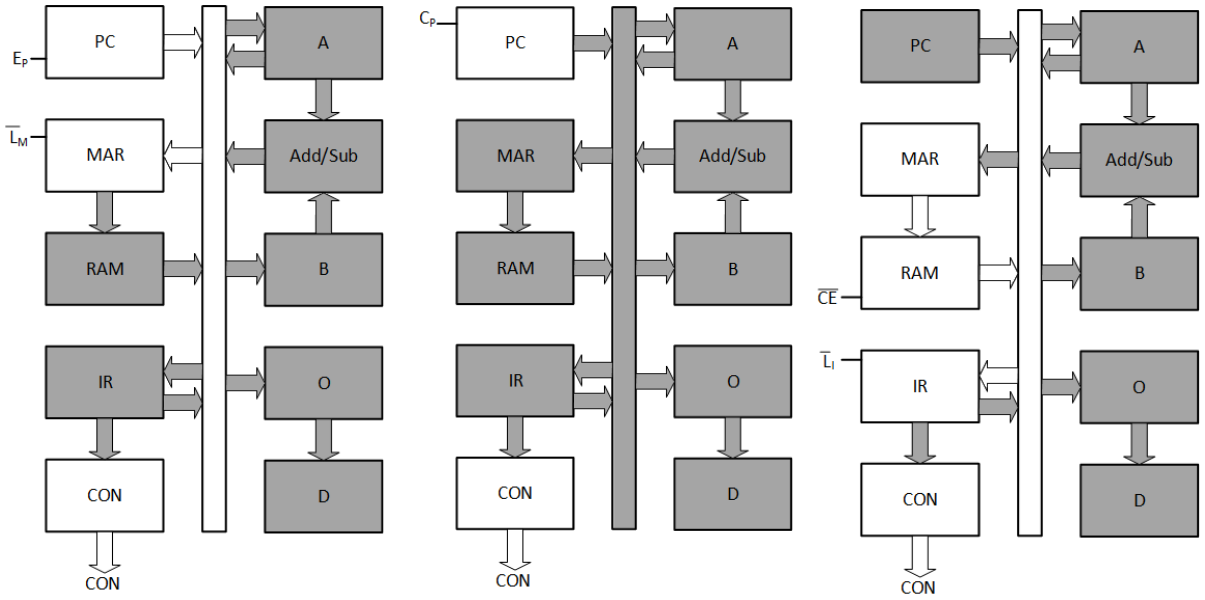
$$\boldsymbol{CON} = C_P E_P \bar{L}_M \overline{CE} \;\; \bar{L}_I \bar{E}_I \bar{L}_A E_A \;\; S_U E_U \bar{L}_B \bar{L}_O$$
$$= 1 \;\; 0 \;\; 1 \;\; 1 \;\;\; 1 \;\; 1 \;\; 1 \;\; 0 \;\;\; 0 \;\; 0 \;\; 1 \;\; 1$$

Kao što vidimo, C_P bit je aktivan.

Memorijsko stanje

T_3 stanje se naziva *memorijsko stanje* zbog toga što se adresirana RAM instrukcija šalje iz memorije u instrukcijski registar. Slika 10-3c prikazuje aktivan deo SAP–1 tokom memorijskog stanja. Jedini aktivni upravljački biti tokom ovog stanja su $\overline{CE}$ i $\bar{L}_I$ i izlazna reč iz kontrolera – sekvencera je:

$$\boldsymbol{CON} = C_P E_P \bar{L}_M \overline{CE} \;\; \bar{L}_I \bar{E}_I \bar{L}_A E_A \;\; S_U E_U \bar{L}_B \bar{L}_O$$
$$= 0 \;\; 0 \;\; 1 \;\; 0 \;\;\; 0 \;\; 1 \;\; 1 \;\; 0 \;\;\; 0 \;\; 0 \;\; 1 \;\; 1$$



a) b) c)

Slika 10-3 – Prihvatni ciklus: (a) stanje T_1; (b) stanje T_2; (c) stanje T_3

Fetch Cycle (Prihvatni ciklus)

Adresno, inkrementalno i memorijsko stanje se nazivaju *prihvatnim ciklusom* SAP–1. Tokom adresnog stanja E_P i $\overline{L}_M$ su aktivni, ovo znači da programski brojač priprema MAR preko W magistrale. Kao što je ranije prikazano na slici 10-2b, pozitivna ivica CLK takta se desi na sredini adresnog stanja, ovo učitava vrednosti programskog brojača u MAR.

C_P je jedini aktivan upravljački bit tokom stanja inkrementa. Ovo podešava programski brojač da broji pozitivne ivice takta. Na sredini stanja inkrementa programski brojač registruje pozitivnu ivicu i brojač se uvećava za jedan.

Tokom memorijskog stanja $\overline{CE}$ i $\overline{L}_I$ su aktivni. Dakle, adresirana RAM reč podešava instrukcijski registar preko W magistrale. Na sredini memorijskog stanja, pozitivna ivica takta učitava adresiranu RAM reč u instrukcijski registar.

10 – 5 Execution Cycle (Izvršni ciklus)

Naredna tri stanja, (T_4, T_5, T_6) pripadaju izvršnom ciklusu SAP–1 sistema. Transferi između registara tokom izvršnog ciklusa zavise od tačno određene instrukcije koja se tad izvršava. Na primer, LDA 9H zahteva različit registarski transfer u odnosu na ADD BH. Ono što sledi su *upravljačke rutine* za različite SAP–1 instrukcije.

LDA rutina

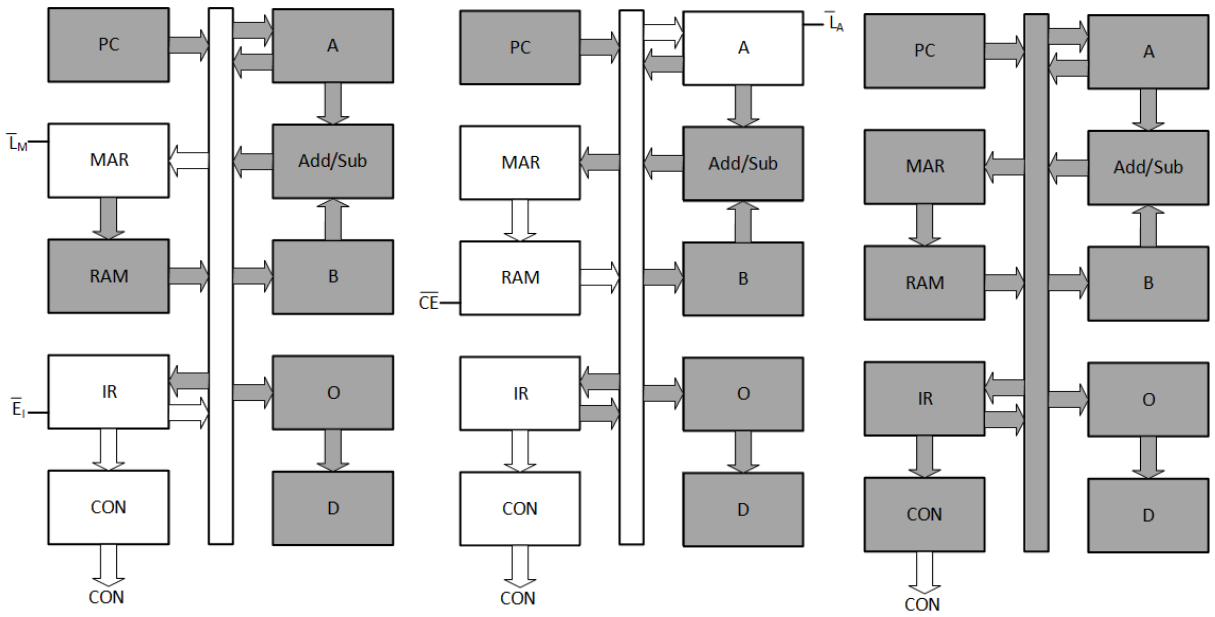
Na konkretnom primaeru, pretpostavimo da je u instrukcijski registar učitana instrukcija LDA 9H.

IR = 0000 1001

Tokom T_4 stanja instrukcijsko polje 0000 se prosleđuje na kontroler – sekvencer gde se dekodira, a adresno polje 1001 se učitava u MAR. Slika 10-4a nam prikazuje aktivne delove SAP–1 tokom T_4 stanja. Obratite pažnju da su $\overline{E}_I$ i $\overline{L}_M$ aktivni, svi drugi upravljački biti su neaktivni.

Tokom T_5 stanja $\overline{CE}$ i $\overline{L}_A$ padaju na logičku nulu. Ovo znači da će adresira reč iz RAM-a učitati u akumulator na sledećoj pozitivnoj ivici takta (pogledati sliku 10-4b).

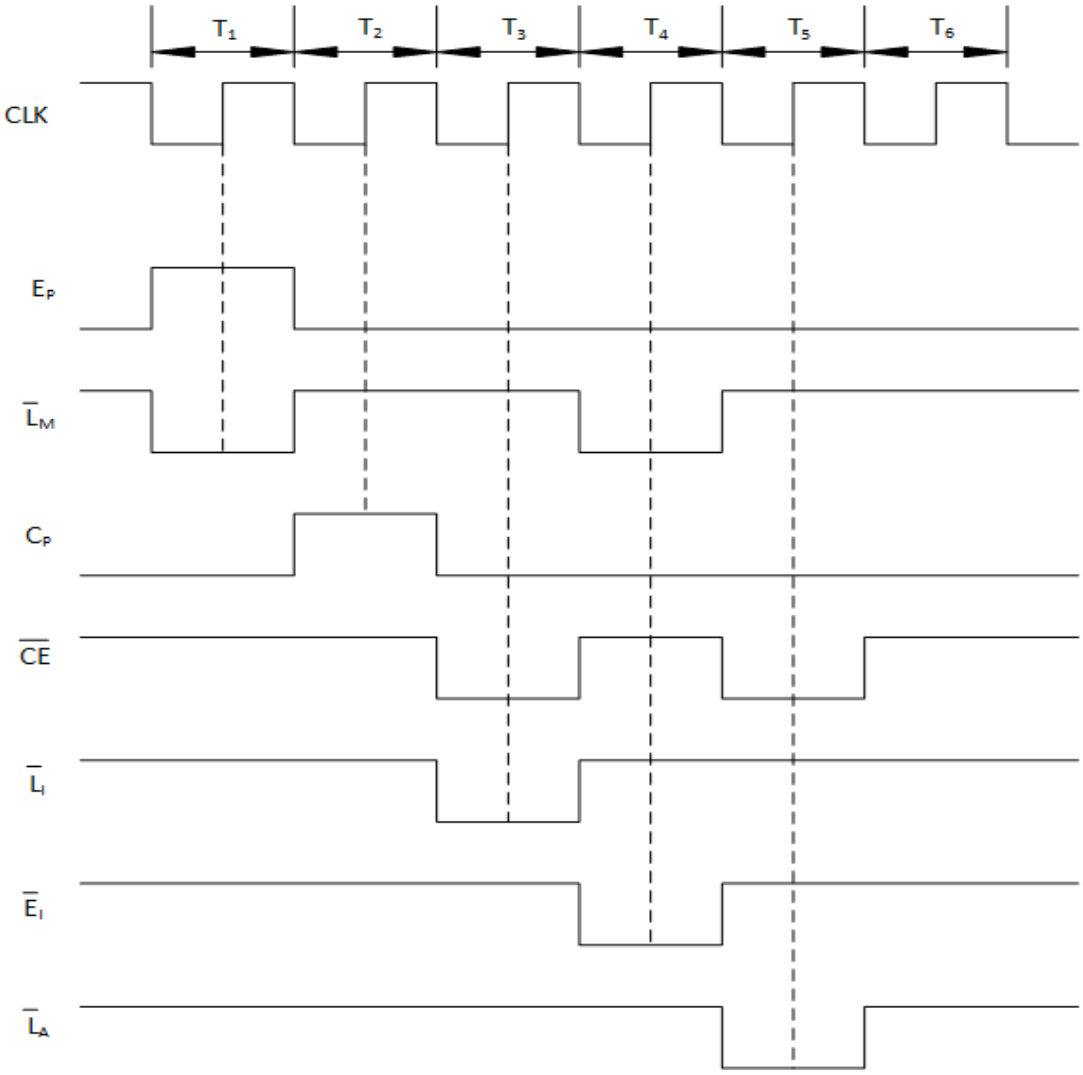
T_6 je stanje bez operacije. Tokom trećeg izvršavanja svi registri su neaktivni (slika 10-4c). Ovo znači da kontroler – sekvence šalje reč sa svim neaktivnim bitima. *Nop* (no operation) znači stanje bez operacije. T_6 stanje LDA rutine je nop.



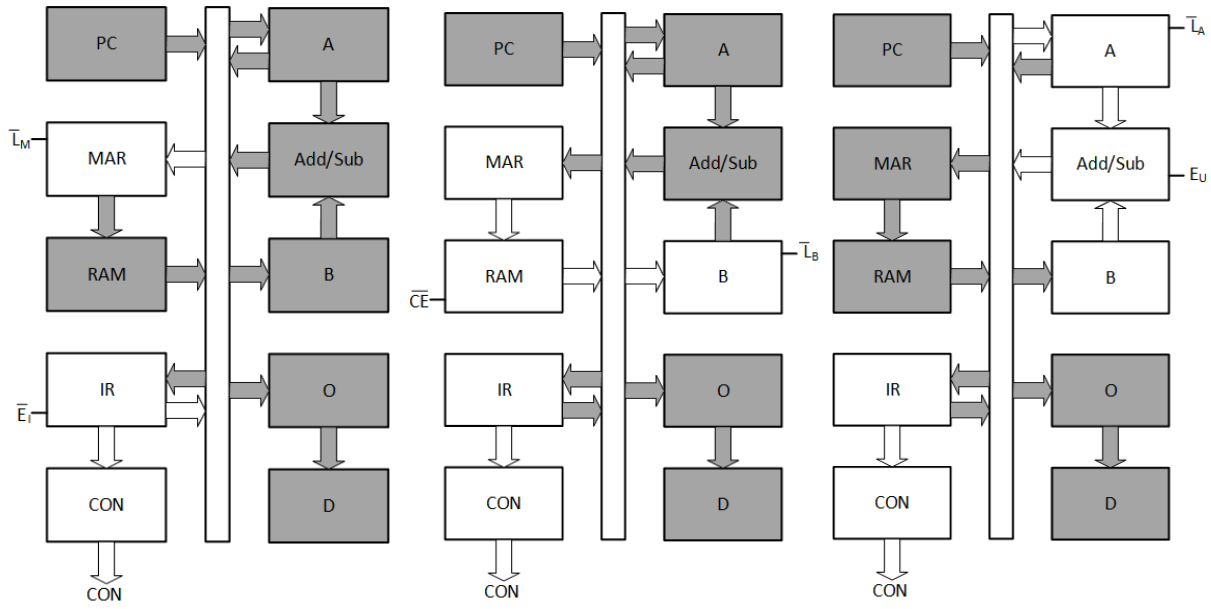
a) b) c)

Slika 10-4 - LDA rutina: (a) stanje T_4; (b) stanje T_5; (c) stanje T_6

Slika 10-5 prikazuje vremenski dijagram za prihvatnu i LDA rutinu. Tokom T_1 stanja E_P i $\overline{L}_M$ su aktivni, pozitivna ivica takta na sredini ovog stanja će proslediti adresu iz programskog brojača u MAR. Tokom T_2 stanja C_P je aktivan i programski brojač se uvećava na pozitivnoj ivici takta. Tokom T_3 stanja $\overline{CE}$ i $\overline{L}_I$ su aktivni. Kada se desi pozitivna ivica takta adresiran RAM reč se prosleđuje u instrukcijski registar. Izvršavanje LDA rutine počinje sa T_4 gde su $\overline{E}_I$ i $\overline{L}_M$ aktivni. Na pozitivnoj ivici takta adresno polje instrukcijskog registra se prosleđuje u MAR. Tokom T_5 stanja $\overline{CE}$ i $\overline{L}_A$ su aktivni što znači da se adresirana RAM reč prosleđuje u akumulator na pozitivnoj ivici takta. Kao što znate, T_6 stanje LDA rutine je nop.



Slika 10-5 – Vremenski dijagram prihvatnog ciklusa i LDA rutine



Slika 10-6 – ADD i SUB rutina: (a) stanje T_4; (b) stanje T_5; (c) stanje T_6

ADD rutina

Pretpostavimo da se na kraju prihvatnog ciklusa u instrukcijskom registru nalazi ADD BH:

$$IR = 0001\ 1011$$

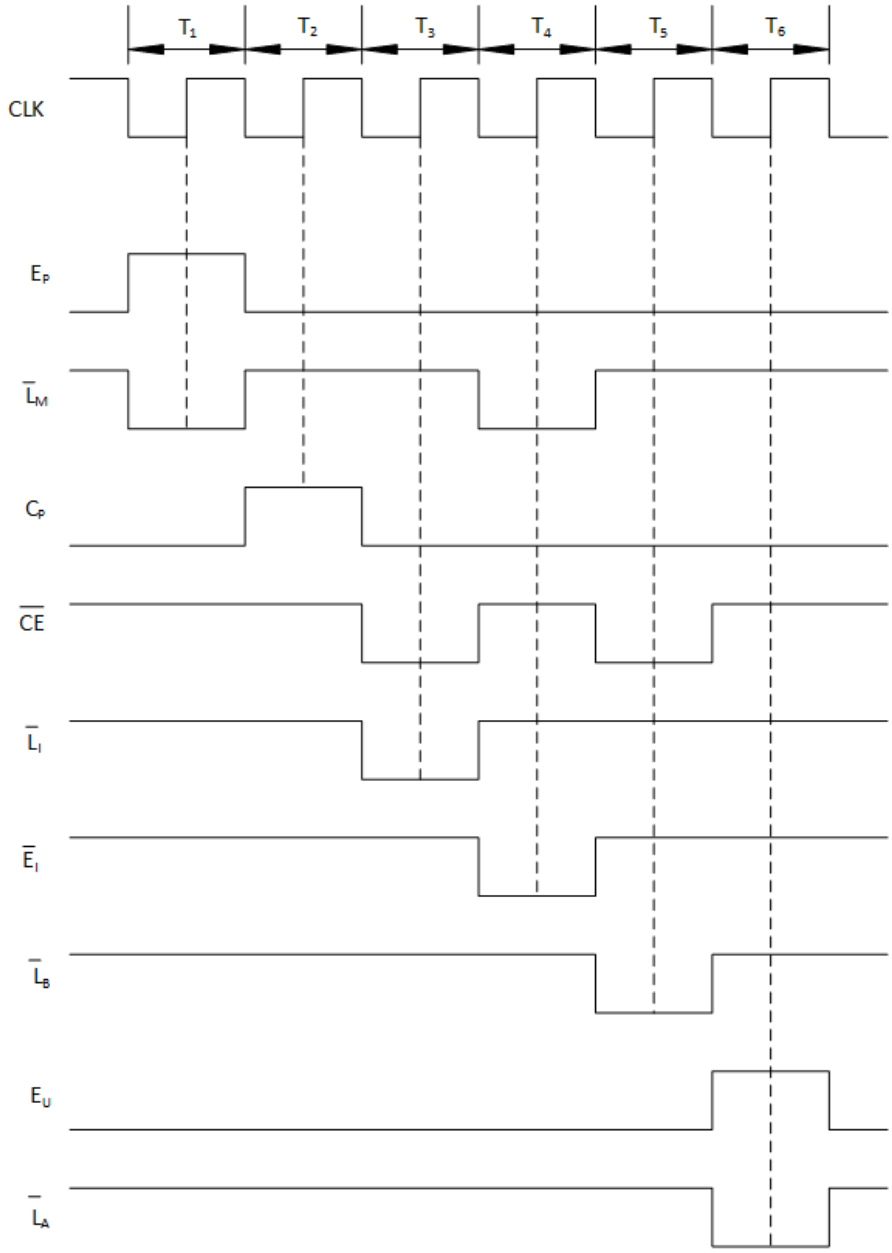
Tokom T_4 stanja instrukcijsko polje se prosleđuje u kontroler – sekvencer, a adresno polje u MAR (slika 10-6a). Tokom ovog stanja $\overline{E}_I$ i $\overline{L}_M$ su aktivni.

Upravljački biti $\overline{CE}$ i $\overline{L}_B$ su aktivni tokom T_5 stanja. Ovo omogućava adresiranoj RAM reči da pripremi B registar (slika 10-6b). Kao i obično, učitavanje se dogodi na sredini stanja, kada CLK ulaz B registra registruje pozitivnu ivicu takta.

Tokom T_6 stanja E_U i $\overline{L}_A$ su aktivni. To znači da sabirač – oduzimač priprema akumulator (slika 10-6c). Na polovini ovog stanja pozitivna ivica takta učitava sumu u akumulator.

Istovremeno, vreme podešavanja i vreme kašnjenja sprečavaju da se akumulator izvrši pre poslednjeg izvršnog stanja. Kada se pojavi pozitivna ivica takta na slici 10-6c sadržaj akumulatora se promeni što prisiljava sabirač – oduzimač da se takođe promeni. Nov sadržaj se vraća na ulaz akumulatora, ali on ne dolazi do tamo sve dok se dese dva signala kašnjenja (jedan za akumulator, jedan sabirač – oduzimač) nakon pozitivne ivice takta. U tom trenutku je prekasno pripremiti akumulator. Ovo sprečava prebrzo izvršavanja akumulatora (učitavanje više od jednog puta na istoj ivici takta).

Slika 10-7 prikazuje vremenski dijagram prihvatne i ADD rutine. Prihvatna rutina je ista kao pre, T_1 stanje učitava adresu programskog brojača u MAR, T_2 stanje uvećava programski brojač, T_3 šalje adresiranu instrukciju u instrukcijski registar. Tokom T_4 stanja $\overline{E}_I$ i $\overline{L}_M$ su aktivni. Na sledećoj pozitivnoj ivici takta adresno polje u instrukcijskom registru ide u MAR. Tokom T_5 stanja $\overline{CE}$ i $\overline{L}_B$ su aktivni, dakle, adresirana RAM reč se učitava u B registar na polovini ovog stanja. Tokom T_6 stanja E_U i $\overline{L}_A$ su aktivni. Kada se desi pozitivna ivica takta suma iz sabirača – oduzimača se smešta u akumulator.



Slika 10-7 – Prihvatni i ADD vremenski dijagram

SUB rutina

SUB rutina je slična ADD rutini. Slike 10-6a i b prikazuju aktivne delove (logičke jedinice) SAP–1 tokom T_4 i T_5 stanja. Tokom T_6 stanja aktivno S_U se šalje u sabirač – oduzimač na slici 10-6c. Vremenski dijagram je skoro identičan kao onaj na slici 10-7. Zamislite neaktivno S_U tokom T_1 i T_5 stanja, a aktivno tokom T_6 stanja.

OUT rutina

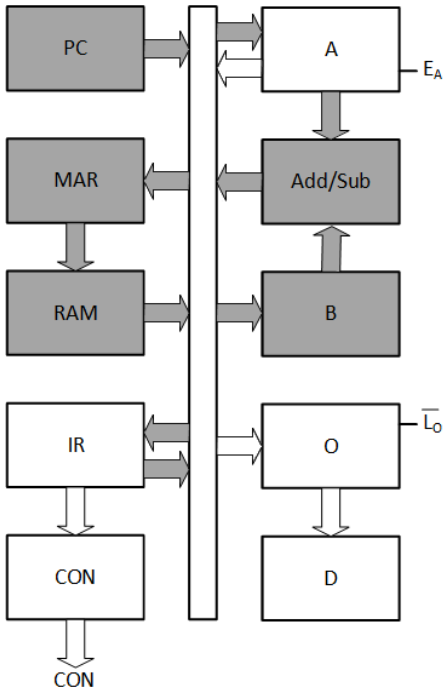
Pretpostavimo da instrukcijski registar sadrži OUT instrukciju na kraju prihvatnog ciklusa. Dakle,

$$IR = 1110\ XXXX$$

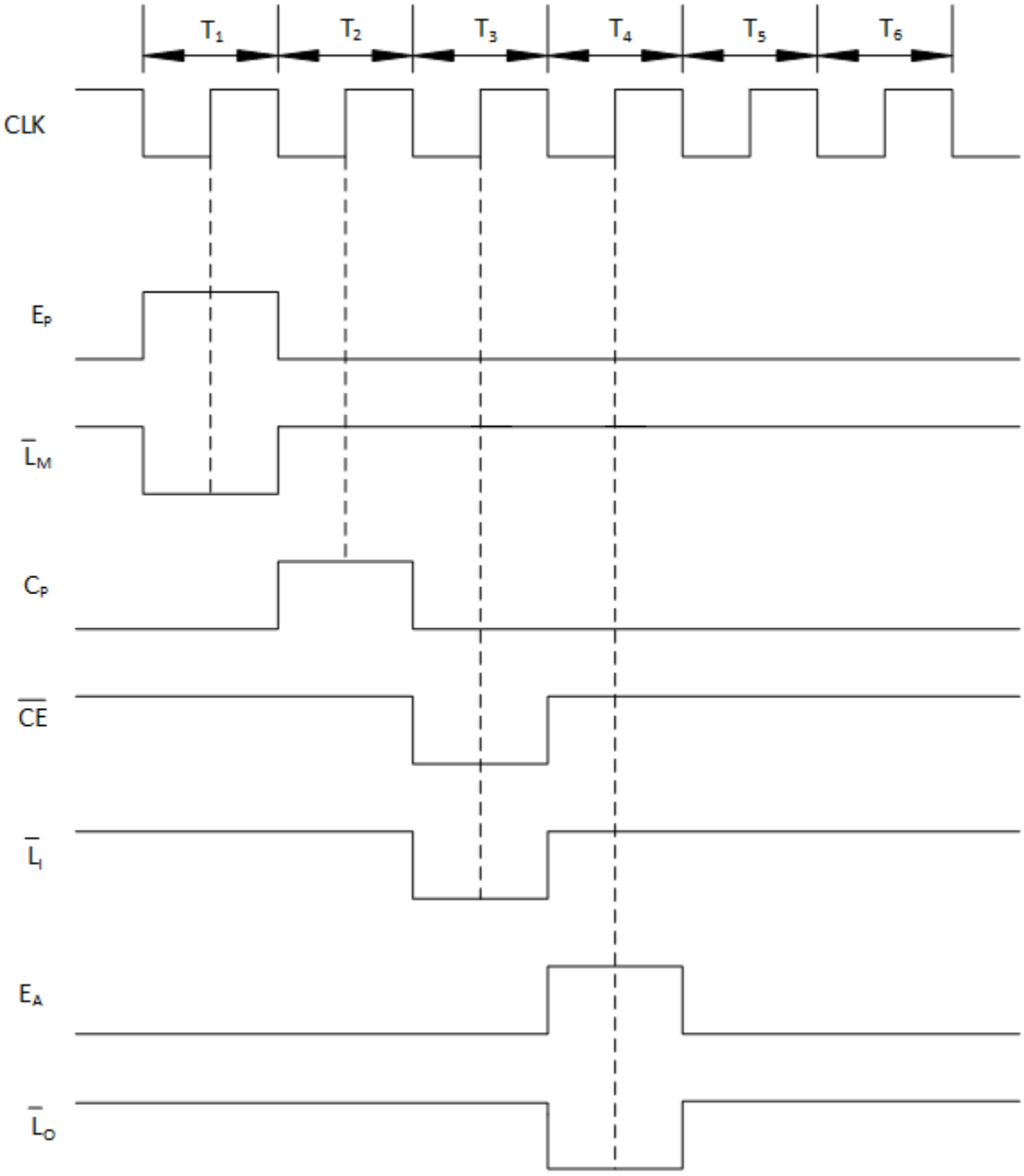
Instrukcijsko polje ide u kontroler – sekvencer da bi se dekodiralo. Zatim, kontroler – sekvencer šalje upravljačku reč potrebnu za učitavanje sadržaja akumulatora u izlazni registar.

Slika 10-8 prikazuje aktivna stanja SAP–1 sistema tokom izvršavanja OUT instrukcije. Pošto su E_A i $\overline{L}_O$ aktivni sledeća pozitivna ivica takta učitava sadržaj akumulatora u izlazni registar tokom T_4 stanja. T_5 i T_6 stanja su nop.

Slika 10-9 je vremenski dijagram prihvatne i OUT rutine. Prihvatna rutina je ista, sadrži adresno, inkrementalno i memorijsko stanje. Tokom T_4 stanja E_A i $\overline{L}_O$ su aktivni, ovo prosleđuje akumulatorsku reč na izlazni registar kada se desi pozitivna ivica takta.



Slika 10-8 – OUT instrukcija (stanje T_4)



Slika 10-9 –Prihvatni i OUT vremenski dijagram

HLT

HLT ne zahteva upravljačku rutinu jer nijedan registar nije uključen u izvršavanje HLT instrukcije. Kada IR ima vrednost:

IR = 1111 XXXX

Instrukcijsko polje 1111 označa kontroleru – sekvenceru da zaustavi procesiranje podataka. Kontroler – sekvencer zaustavlja računar isključivanjem takta (clock, biće objašnjeno kasnije).

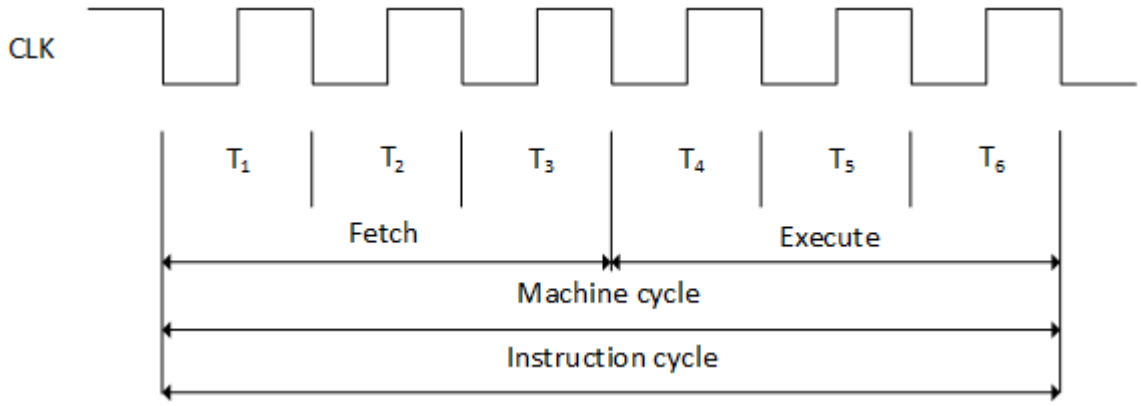
Machine Cycle and Instruction Cycle (Mašinski i instrukcijski ciklus)

SAP–1 ima šest T stanja (tri prihvatna i tri izvršna). Ovih šest stanja se nazivaju *mašinski ciklus* (slika 10-10a). Potreban je jedan mašinski ciklus da bi se prihvatila i izvršila jedna instrukcija. SAP-1 ima takt čija je frekvencija 1 kHz što je ekvivalentno periodi od 1 ms. Dakle, potrebno je 6 ms za jedan mašinski ciklus SAP–1.

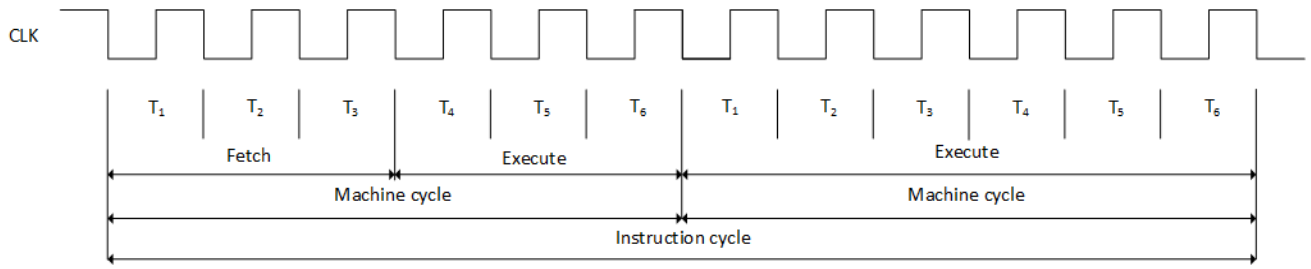
SAP–2 se malo razlikuje zbog toga što neke instrukcije zahtevaju više od jednog mašinskog ciklusa da bi se prihvatile i izvršile. Slika 10-10b prikazuje vremenski dijagram za instrukciju koja zahteva dva mašinska ciklusa. Prva tri stanja su prihvatnog ciklus, ali izvršni ciklus zahteva narednih 9 T stanja. Ovo je zbog toga što je instrukcija koja zahteva dva mašinska ciklus komplikovanija i zahteva dodatna T stanja da bi se izvršila.

Broj T stanja koja su potrebna za prihvatanje i izvršavanje instrukcija nazivaju se *instrukcijski ciklus*. U SAP–1 instrukcijski ciklus je identičan mašinskom ciklusu dok u SAP-2 i drugim mikroračunarima može biti jedan dva ili više mašinskih ciklusa, što je prikazano na slici 10-10b.

Instrukcijski ciklus za 8080 i 8085 imaju od jedan do pet mašinskih ciklusa.



Slika 10-10a – SAP-1 instrukcijski ciklus



Slika 10-10b – Instrukcijski ciklus sa dva mašinska ciklusa

Primer 10-5

Uputstvo za programiranje 8080/8085 navodi da je potrebno trinaest T stanja za prihvatanje i izvršavanje LDA instrukcije. Ako sistemski takt imam frekvenciju 2.5 MHz koliko dugo traje instrukcijski ciklus?

Rešenje

Perioda taktnog signala je:

$$T = \frac{1}{f} = \frac{1}{2.5\ MHz} = 400\ ns$$

Dakle, svako T stanje traje 400 ns. Pošto je potrebno trinaest T stanja za prihvatanje i izvršavanje LDA instrukcije instrukcijski ciklus traje

$$13 \text{ x } 400 \text{ ns} = 5.200 \text{ ns} = 5{,}2\ \mu s$$

Primer 10-6

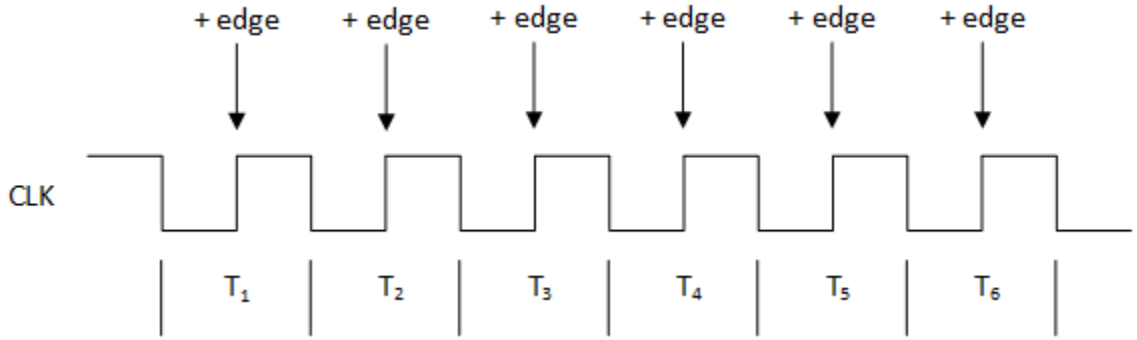
Slika 10-11 prikazuje šest T stanja SAP–1. Pozitivna ivica takta se pojavljuje na polovini svakog stanja. Zašto je ovo važno?

Rešenje

SAP–1 je *magistralno organizovan računar* (kao i većina današnjih računara). Ovo omogućava njegovim registrima da komuniciraju preko W magistrale. Ali do pouzdanog učitavanja registara dolazi samo kad je vreme za podešavanje i vreme za čekanje pravilno postavljeno. Čekanje pola ciklusa pre učitavanja registra zadovoljava uslov vezan za vreme podešavanja, čekanje pola ciklusa nakon učitavanja registra zadovoljava uslov vezan za vreme čekanja. Zbog ovoga je pozitivna ivica takta dizajnirana tako da se pojavi na sredini svakog T stanja (slika 10-11).

Postoji još jedan razlog zbog kojeg se čeka pola ciklusa pre učitavanja registra. Kad ENABLE ulaz registra koji šalje podatke postane aktivan sadržaj registra se odjednom

pošalje na W magistralu. Parazitivna kapacitivnost i induktivnost sprečavaju linije magistrale da odmah dođu do svog tačno određenog nivoa napona. Drugim rečima, dobijamo privremene varijacije u naponu na magistrali i moramo da čekamo da one nestanu da bismo bili sigurni da imamo ispravne podatke u trenutku učitavanja. Čekanje od pola ciklusa omogućava podacima da budu ispravni pre učitavanja.



Slika 10-11 – Pozitivna ivica takta koja se pojavljuje sredinom T stanja

---

10-6 SAP-1 mikroprogram

Uskoro ćemo analizirati šematski dijagram SAP–1 računara, ali prvo moramo da sumiramo izvršavanje SAP–1 instrukcija u tabelu koja se zove *mikroprogram*.

Mikroinstrukcije

Kontroler – sekvencer šalje upravljačke reči, po jednu u svakom T stanju ili ciklusu takta. Ove reči su kao uputstva koja govore ostatku računara šta da rade. Pošto to predstavlja samo mali korak u obradi podataka svaka upravljačka reč se zove *mikroinstrukcija*. Kada se pogleda na SAP–1 blok dijagram (slika 10-1) možemo da zamislimo konstantan tok mikroinstrukcija koje iz kontrolera – sekvencera idu kao ostalim komponentama SAP–1 računara.

Makroinstrukcije

Instrukcije koje smo koristili za programiranje (LDA, ADD, SUB, ...) se ponekad zovu *makroinstrukcije* da bismo ih razlikovali od mikroinstrukcija. Svaka SAP–1 makroinstrukcija se sastoji od tri mikroinstrukcije. Na primer, LDA makroinstrukcija se

sastoji od mikroinstrukcija koje se nalaze u tabeli 10-3. Da bismo uprostili pojavi ovih mikroinstrukcija možemo da koristimo heksadecimalni prikaz kao u tabeli 10-4.

Tabela 10-5 prikazuje SAP–1 mikroprogram i listu svake makroinstrukcije i mikroinstrukcije koja je potrebna za njegovo izvršavanje. Ova tabela sumira izvršne rutine za SAP–1 instrukcije. Slična tabela može da se koristi i za naprednije instrukcijske setove.

Tabela 10-3

Makro	**Stanje**	**C_P E_P $\bar{L}_M$ $\overline{CE}$**	**$\bar{L}_I$ $\bar{E}_I$ $\bar{L}_A$ E_A**	**S_U E_U $\bar{L}_B$ $\bar{L}_O$**	**Aktivno**
LDA	T_4	0001	1010	0011	$\bar{L}_M$, $\bar{E}_I$
	T_5	0010	1100	0011	$\overline{CE}$, $\bar{L}_A$
	T_6	0011	1110	0011	Nijedan

Tabela 10-4

Makro	**Stanje**	**CON**	**Aktivno**
LDA	T_4	1A3H	$\bar{L}_M$, $\bar{E}_I$
	T_5	2C3H	$\overline{CE}$, $\bar{L}_A$
	T_6	3E3H	Nijedan

Tabela 10-5 – SAP-1 mikroprogram

Makro	**Stanje**	**CON**	**Aktivno**
LDA	T_4	1A3H	$\bar{L}_M$, $\bar{E}_I$
	T_5	2C3H	$\overline{CE}$, $\bar{L}_A$
	T_6	3E3H	Nijedan
ADD	T_4	1A3H	$\bar{L}_M$, $\bar{E}_I$
	T_5	2E1H	$\overline{CE}$, $\bar{L}_B$
	T_6	3C7H	$\bar{L}_A$, E_U
SUB	T_4	1A3H	$\bar{L}_M$, $\bar{E}_I$
	T_5	2E1H	$\overline{CE}$, $\bar{L}_B$
	T_6	3CFH	$\bar{L}_A$, S_U, E_U
OUT	T_4	3F2H	E_A, $\bar{L}_O$
	T_5	3E3H	Nijedan
	T_6	3E3H	Nijedan

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

10 – 7 SAP–1 šematski dijagram

U ovoj sekciji proučavamo kompletan šematski dijagram za SAP–1. Slike 10-12 do 10-15 prikazuju sve čipove, žice i signale. Trebalo bi obratiti pažnju na ove slike tokom ovog predavanja. Dodatak 4 daje dodatne detalje za neke od komplikovanijih čipova.

Program Counter (Programski brojač)

Čipovi C1, C2 i C3 sa slike 10-12 su *programski brojači*. Čip C1 je 74LS107, dvostruki JK master – slave flip – flop koji kreira 2 viša adresna bita. Čip C2, još jedan 74LS107, kreira 2 niža adresna bita. Čip C3 je 74LS126, četvorostruki normalno otvoreni switch sa tri stanja. On daje programskom brojaču tri stanja na izlazu.

Na početku rada računara, $\overline{CLR}$ koji je na nuli resetuje programski brojač na 0000. Tokom T_1 stanja E_P, koje je na logičkoj jedinici, smešta adresu na W magistralu. Tokom T_2 stanja C_P, koje je na logičkoj jedinici, se predaje programskom brojaču, na sredini ovog stanja negativna ivica $\overline{CLK}$ signala (ekvivalentna pozitivnoj ivici CLK signala) uvećava programski brojač.

Programski brojač je neaktivan u stanjima od T_3 do T_6.

MAR

Čip C4, 74LS173, je četvorobitni bafer registar, služi kao MAR. Obratiti pažnju da su pinovi 1 i 2 uzemljeni što konvertuje izlaz sa tri stanja u izlaz sa dva stanja. Drugim rečima, izlaz MAR-a nije povezan na W magistralu pa tako i ne postoji potreba za izlazom sa tri stanja.

2–to–1 Multiplexer (2 prema 1 multiplekser)

Čip C5 je 74LS157, 2 prema 1 nibble (4 bita) *multiplekser*. Leva četiri bita (pinovi 14, 11, 5, 2) dolaze iz adresnog switch registra (S_1). Desna četiri bita (pinovi 13, 10, 6, 3) dolaze od MAR-a. RUN-PROG switch (S_2) bira odgovarajućih četiri bita da bi došao do izlaza C5 čipa. Kada je S_2 u PROG poziciji selektovana su četiri bita adresnog switch registra. Sa druge strane, kada je S_2 u RUN poziciji izabran je izlaz iz MAR-a.

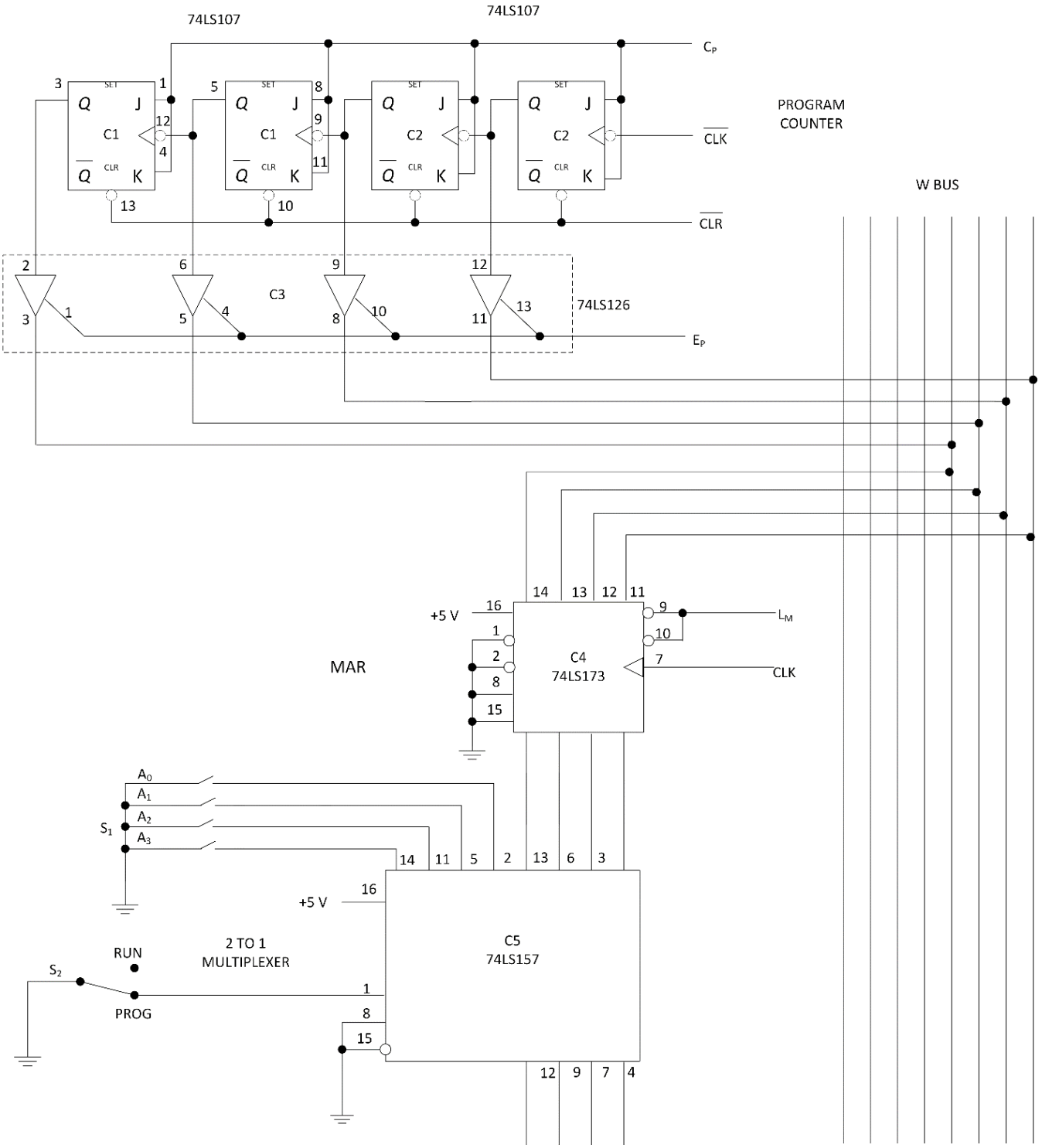
16 × 8 RAM

Čipovi C6 i C7 su 74189. Oba čipa su 16 × 4 statički RAM. Zajedno čine 16 × 8 *upisno – čitajuću* (read – write) *memoriju*. S_3 je osmobitni data switch registar, a S_4 je upisno – čitajući switch („push – button switch“). Da bi se programirala memorija S_2 se postavi u PROG poziciju, ovo spusti $\overline{CE}$ ulaz na nulu (pin 2). Kratkotrajan impuls na upisno – čitajući switch obara $\overline{WE}$ ulaz na nulu (pin 3) i učitava memoriju.

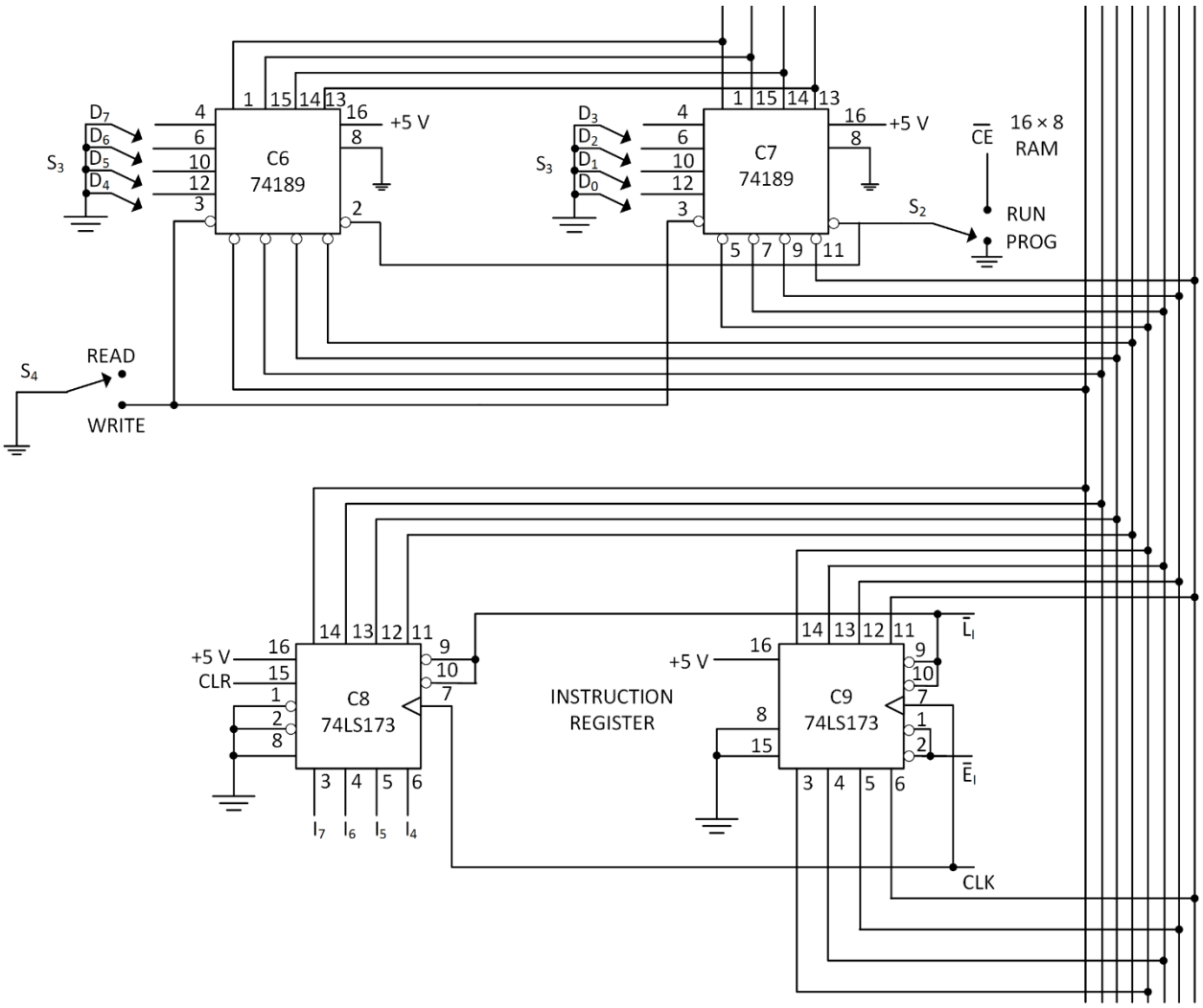
Nakon što su program i podaci u memoriji, RUN – PROG switch (S_2) se postavi u RUN poziciju i računar se priprema za rad.

Instrukcijski registar

Čipovi C8 i C9 su 74LS173s. Oba čipa su četvorobitni bafer registri sa tri stanja. Oni su *instrukcijski registri*. Uzemljeni pinovi 1 i 2 C8 čipa konvertuju izlaz sa tri stanja u izlaz sa dva stanja, $I_7I_6I_5I_4$. Ova četiri bita idu na instrukcijski dekoder u kontroleru – sekvenceru. Signal $\overline{E_I}$ kontroliše izlaz C9 tj. niža četiri bita u instrukcijskom registru. Kada je $\overline{E_I}$ na nuli ova četiri bita su smešteni na W magistralu.

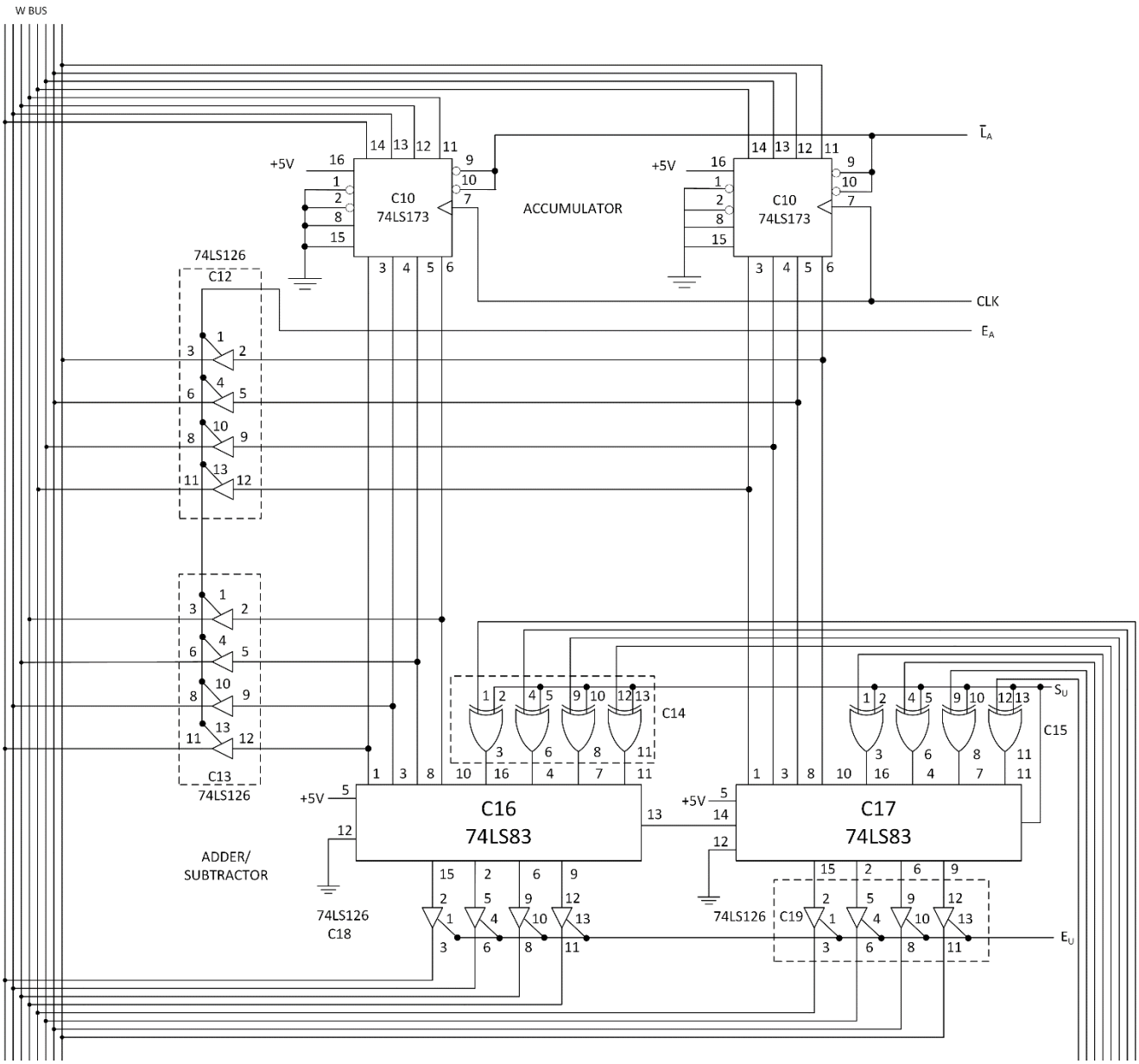

74LS107
74LS107
C_P
PROGRAM COUNTER
$\overline{CLK}$
$\overline{CLR}$
W BUS
C3
74LS126
E_P
MAR
+5 V
C4
74LS173
L_M
CLK
A_0
A_1
A_2
A_3
S_1
+5 V
C5
74LS157
RUN
2 TO 1
MULTIPLEXER
S_2
PROG

a)

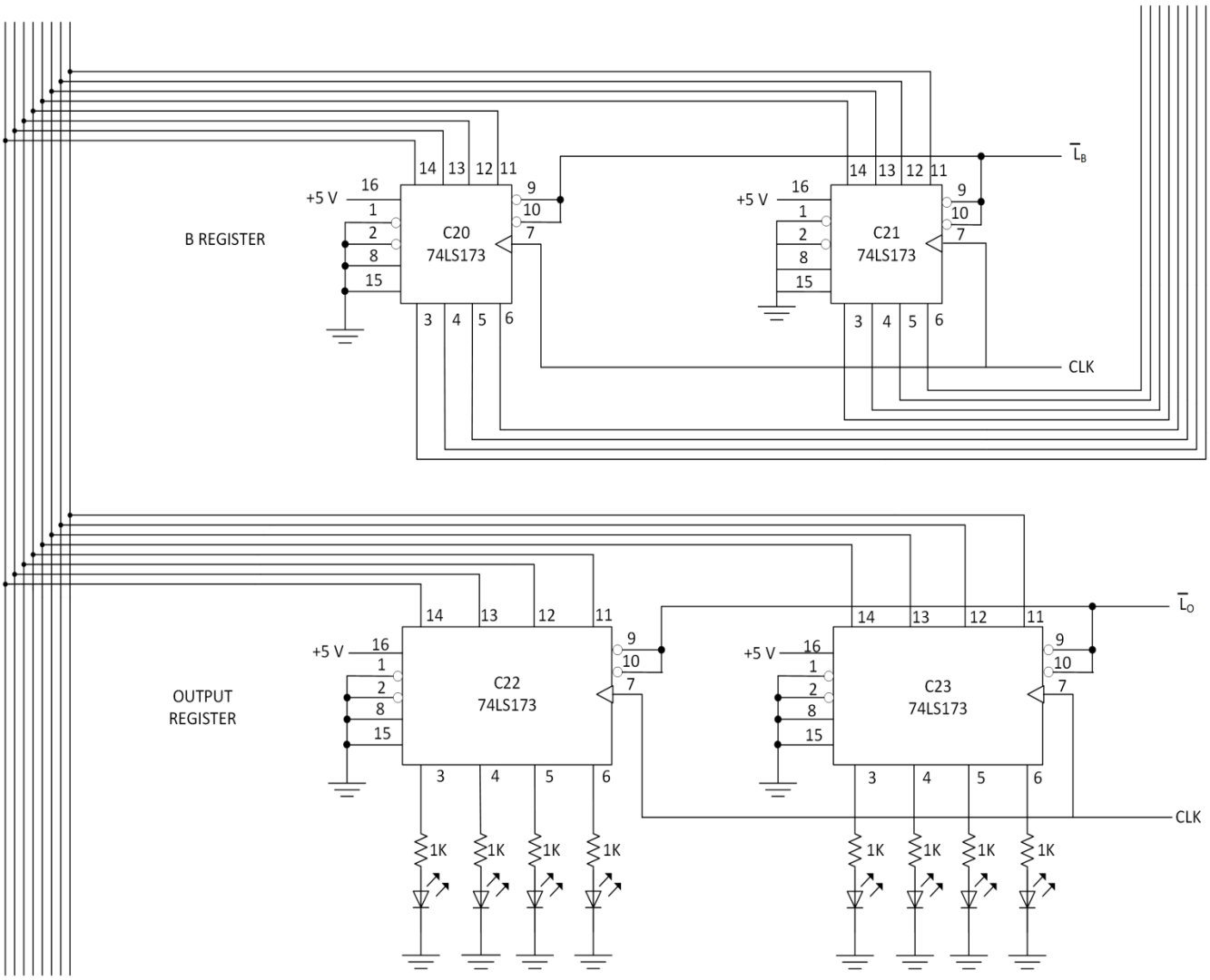


b)

Slika 10-12 (a)(b) – Programski brojač, memorija i instrukcijsi registar


W BUS
+5V
C10
74LS173
ACCUMULATOR
$\overline{L}_A$
CLK
E_A
74LS126
C12
C13
74LS126
C14
C15
S_U
C16
74LS83
C17
74LS83
ADDER/
SUBTRACTOR
74LS126
C18
74LS126
C19
E_U

a)



b)

Slika 10-13 (a)(b) – A i B registri, sabirač – oduzimač i izlazna kola

Akumulator

Čipovi C10 i C11 , 74LS173s, su *akumulatori* (slika 10-13), Pinovi 1 i 2 su uzemljeni na oba čipa da bismo imali dva stanja na izlazu sabirača – oduzimača. Čipovi C12 i C13 su 74LS126, switch sa tri stanja smešta vrednost akumulatora na W magistralu kada je E_A na logičkoj jedinici.

Adder – Substracter (Sabirač – Oduzimač)

Čipovi C14 i C15 su 74LS86. Ova ekskluzivna – ili kola su kontroleri invertori. Kada je S_U na logičkoj nuli vrednosti B registra se šalje. Kada je S_U na logičkoj jedinici prvi komplement se šalje, a jedinica se dodaje LSB-u da bi se formirao drugi komplement.

Čipovi C16 i C17 su 74LS83. Ovaj četvorobitni potpuni sabirači se kombinuju da naprave osmobitnu sumu ili razliku. Čipovi C18 i C19 su 74LS126, oni konvertuju osmobitno rešenje u stanje sa tri izlaza za kontrolu W magistrale.

B register and Output Register (B registar i izlazni registar)

Čipovi C20 i C21 su 74LS173, oni formiraju *B registar*. Sadrže podatke koji će biti dodati ili oduzeti od akumulatora. Uzemljeni pinovi 1 i 2 na oba čipa obezbeđuju izlaz sabirača - oduzimača sa dva stanja.

Čipovi C22 i C23 su 74LS173, oni formiraju izlazni registar. Oni upravljaju binarnim displejem i omogućavaju nam da vidimo obrađene podatke.

Clear – Start Debouncer

Na slici 10-14 *clear – start debouncer* obezbeđuje dva izlaza, CLR za instrukcijski registar i $\overline{\text{CLR}}$ za programski brojač i kružni brojač. $\overline{\text{CLR}}$ takođe ide na C29, clock – start flip – flop, S5 je push – button switch. Kada nije pritisnut prebacuje se u CLEAR poziciju i generiše logičku jedinicu na CLR i nulu na. $\overline{\text{CLR}}$. Kada je S_5 otpušten on vraća Start poziciju i generiše logičku nulu na CLR i jedinicu na $\overline{\text{CLR}}$.

Obratite pažnju da se polovina C24 koristi za clear – start debouncer, a druga polovina za single – step debouncer. Čip C24 je 7400, četvorostruko NI kolo sa dva ulaza.

Single – Step Debouncer

SAP–1 može da radi u dva režima, ručnom i automatskom. U ručnom režimu se pritisne i otpusti S_6 da bi se dobio jedan taktni impuls. Kada je S_6 pritisnut CLK takt je na logičkoj jedinici, kada je otpušten onda je na logičkoj nuli. Drugim rečima, *single – step debouncer* sa slike 10-14 generiše T stanja jedno po jedno, prilikom svakog pritiskanja switch-a. Ovo omogućava prolazak kroz različita T stanja korak po korak dok se pretražuju i otklanjaju greške (troubleshooting and debugging) u softveru ili hardveru.

Manual – Auto Debouncer (Ručni – Automatski debouncer)

Switch S_7 je „single – pole double – throw“ (SPDT) switch i može da ostane u ručnom ili automatskom režimu rada. Kada je prekidač namešten u poziciju MANUAL tada je aktivno single – step dugme. Kada je u poziciji AUTO računar radi automatski. Dva logička NI kola u čipu C26 čine deo NI – NI mreže koja usmerava single – step takt ili automatski takt ka CLK i $\overline{CLK}$ izlazima.

Clock Buffers (Baferi takta)

Izlaz sa pina 11, čip C26, upravlja *baferima takta*. Kao što se vidi na slici 10-14 dva invertora se koriste da bi proizveli konačni CLK izlaz i jedan invertor da proizvede $\overline{CLK}$ izlaz. Za razliku od većine čipova, C27 je standardni TTL, a ne Schottky niske snage (low – power Schottky) (pogledajte SAP–1 listu sa delovima, dodatak 5). Standardni TTL se koristi jer on može da upravlja sa 20 Schottky TTL potrošača niske snage, kao što se vidi u tabeli 4-5.

Ako proverite tabelu sa podacima i ulazne struje za 74LS107 i 74LS173 moći ćete da prebrojite sledeća low – power Schottky (LS) TTL opterećenja na taktnim i CLR signalima:

$$\begin{aligned} CLK &= 19 \text{ LS potrošača} \\ \overline{CLK} &= 2 \text{ LS potrošača} \\ CLR &= 1 \text{ LS potrošač} \\ \overline{CLR} &= 20 \text{ LS potrošač} \end{aligned}$$

To znači da su CLK i $\overline{CLK}$ signali iz C27 čipa (standardni TTL) su adekvatni za upravljanje Schottky TTL potrošačima niske snage. Takođe, CLR i $\overline{CLR}$ signali iz C24 čipa (standardni TTL) mogu da upravljaju tim potrošačima.

Clock Circuits and Power Supply (Kola za takt i jedinica za napajanje)

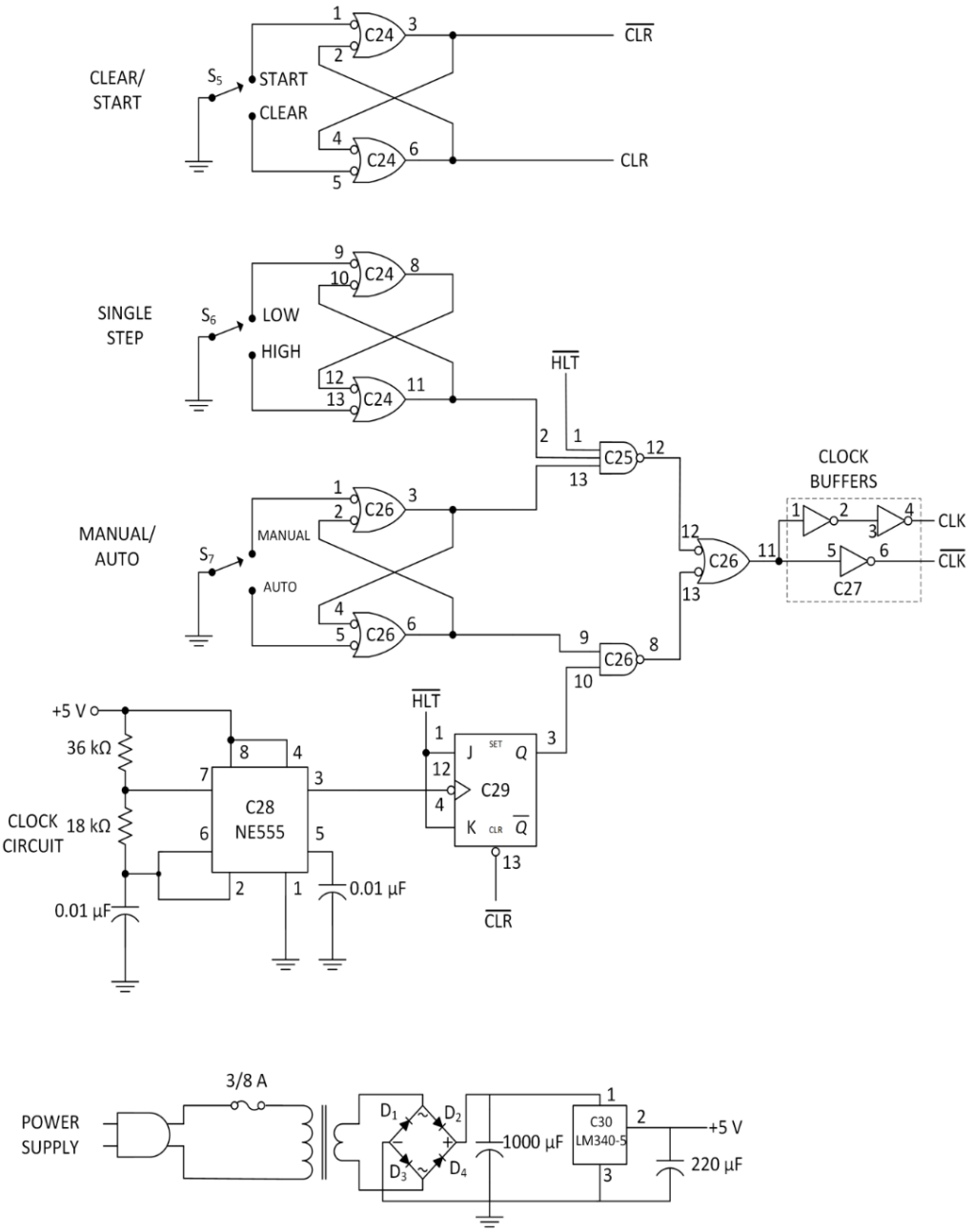
Čip C28 je 555 tajmer. Ovo integrisano kolo proizvodi pravougaoni signal frekvencije 2 kHz i koji ima 75% duty cycle (procenat periode za koju je signal na logičkoj jedinici). Kao što smo ranije diskutovali, *flip – flop za pokretanja takta* („start – the – clock flip – flop), čip C29, deli signal na 1 kHz i istovremeno proizvodi 50% duty cycle.

Jedinica za napajanje sastoji se od dvostranog ispravljača („full – wave“ bridge rectifier) sa kapacitivnim filtrom. Jednosmerni napon na kondenzatoru od 1000μF iznosi otprilike 20 V. Čip C30, LM340T-5 je naponski regulator koji proizvodi stabilan izlaz od +5 V.

Instrukcijski dekoder

Čip 31, heksadecimalni invertor, proizvodi komplemente bitova ($I_7I_6I_5I_4$, slika 10-15) operacionog koda. Zatim čipovi C32, C33 i C34 dekodiraju operacioni kod i kreiraju pet izlaznih signala: LDA, ADD, SUB, OUT i $\overline{\text{HLT}}$. Samo jedan od njih je aktivan u nekom trenutku ($\overline{\text{HLT}}$ je aktivan kad je na logičkoj nuli, ostali su aktivni kad su na logičkoj jedinici).

Kada je HLT instrukcija u *instrukcijskom registru* bitovi $I_7I_6I_5I_4$ su 1111 i $\overline{\text{HLT}}$ je na nuli. Ovaj signal se vraća u čipove C25 (single – step takt) i C29 (automatski takt). U oba režima rada, MANUAL ili AUTO, takt se zaustavlja i računar prestaje sa radom.

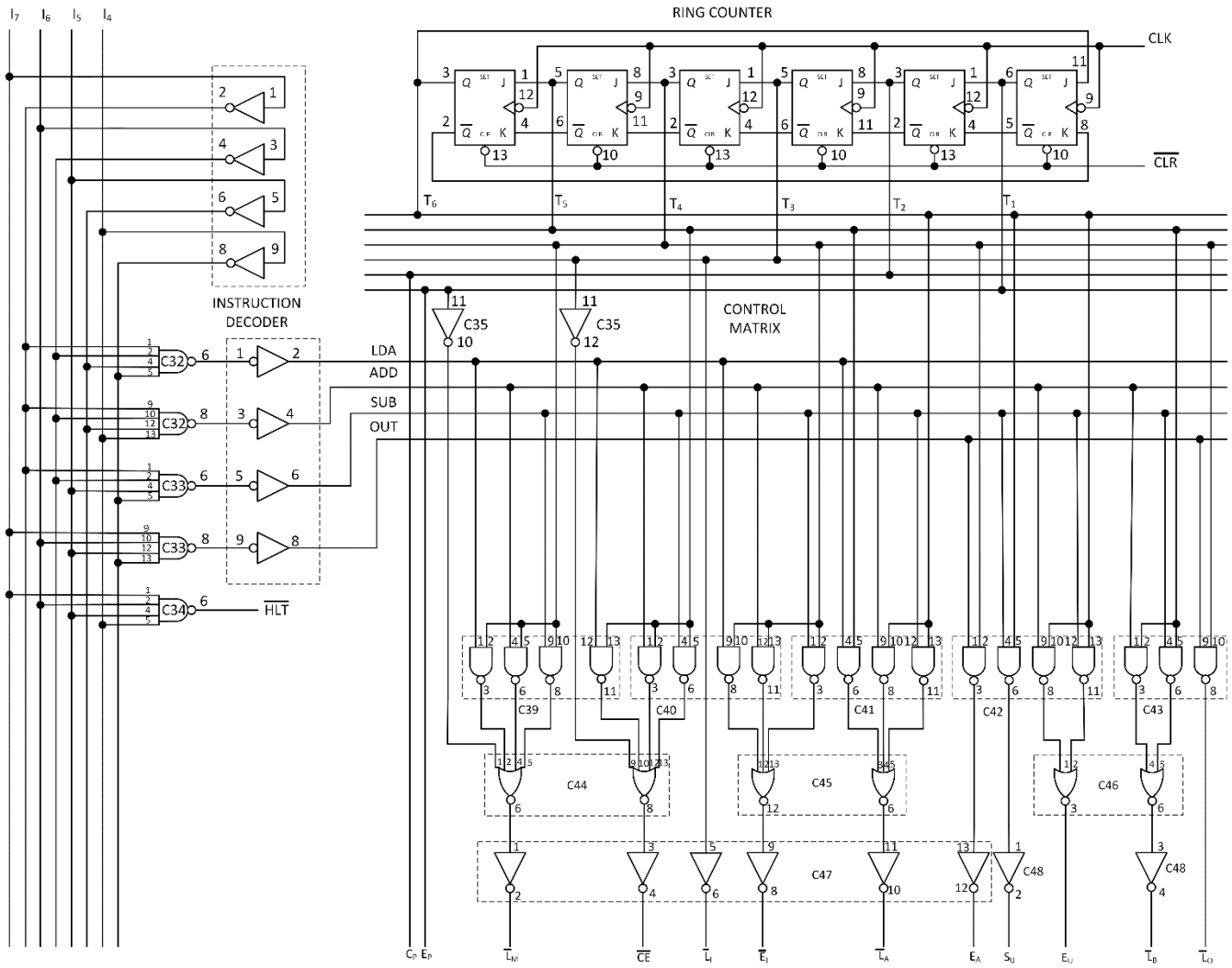


Slika 10-14 – Napajanje, takt (CLK) i CLR kola

Ring Counter (Kružni brojač)

Kružni brojač, nekad nazivan i *brojač stanja*, sastoji se od tri čipa, C36, C37 i C38. Svaki od ovih čipova je 74LS107, dvostruki JK master – slave flip – flop. Ovaj brojač se resetuje kada se pritisne clear – start dugme (S_5). Izlaz Q_0 flip – flopa je invertovan tako da njegov izlaz $\overline{Q}$ (pin 6, C38) upravlja J ulazom Q_1 flip – flopa (pin 1, C38). Zbog ovoga je T_1 izlaz na logičkoj jedinici na početku.

CLK signal upravlja aktivnim ulazom na logičkoj nuli. Ovo znači da negativna ivica CLK signala pokreće svako T stanje. Pola ciklusa kasnije, pozitivna ivica CLK vrši učitavanje registara, kao što je ranije opisano.



Slika 10-15 – Instrukcijski decoder, kružni brojač i upravljač matrice

Control Matrix (Upravljačka matrica)

LDA, ADD, SUB i OUT signali iz instrukcijskog dekodera upravljaju *upravljačkom matricom* koja se sastoji od čipova C39 do C48. Istovremeno, signali iz kružnog brojača, od T_1 do T_6 upravljaju matricom (kolo koje prima dve grupe signala sa dva različita izvora). Matrica kreira CON, dvanaestobitnu mikroinstrukciju koja govori ostatku računara šta da radi.

Na slici 10-15, T_1 je na logičkoj jedinici, zatim T_2 pa T_3 itd. Ako analizirate upravljačku matricu videćete da T_1 na logičkoj jedinici stvara E_P na logičkoj jedinici i $\bar{L}_M$ (adresno stanje) na logičkoj nuli, T_2 na jedinici stvara C_P (stanje inkrementa) na jedinici, a T_3 na jedinici stvara $\overline{CE}$ i $\bar{L}_I$ (memorijsko stanje) na nuli. Prva tri T stanja, dakle, su uvek deo prihvatnih ciklusa SAP–1 računara. CON reči za prihvatni ciklus su

Stanje	CON	Aktivni biti
T_1	5E3H	E_P, $\bar{L}_M$
T_2	BE3H	C_P
T_3	263H	$\overline{CE}$, $\bar{L}_I$

Tokom izvršnih stanja, T4, T5 i T6 idu na logičku jedinicu jedan za drugim. Istovremeno, samo jedan od dekodiranih signala (od LDA do OUT) je na jedinici. Zbog ovoga matrica automatski vodi aktivne bite do pravih izlaznih upravljačkih linija.

Na primer, kada je LDA na jedinici jedina aktivirana NI kola sa dva ulaza su prvo, četvrto, sedmo i deseto kolo. Kada je T_4 na jedinici aktiviraju se prvo i sedmo NI kolo što rezultuje sa $\bar{L}_M$ i E_I (učitava adresna polja u MAR) na logičkoj nuli. Kada je T5 na logičkoj jedinici aktiviraju se četvrto i deseto NI kolo što dovodi do logičke nule na $\overline{CE}$, i $\bar{L}_A$ (učitava RAM podatke u akumulator). Kada je T_6 na logičkoj jedinici nijedan od upravljačkih bita nije aktivan (nop).

Trebalo bi analizirati akcije upravljačke matrice tokom izvršnih stanja za sledeće slučajeve: ADD na jedinici, SUB na jedinici, OUT na jedinici. Tada biste se složili da upravljačka matrica može da generiše ADD, SUB i OUT mikroinstrukcije prikazane u tabeli 10-15 (SAP–1 mikroprogram).

Operacija

Pre svakog pokretanja računara operater unosi program i podatke u SAP–1 memoriju. Kada je program u nižoj, a podaci u višoj memoriji operater pritiska i otpušta clear dugme. CLK i $\overline{CLK}$ signali upravljaju registriima i brojačima. Mikroinstrukcija iz kontrolera – sekvencera određuje šta će se desiti na svakoj pozitivnoj CLK ivici.

Svaki SAP–1 mašinski ciklus počinje prihvatnim ciklusom, T1 je adresno stanje, T2 je stanje inkrementa, a T3 je memorijsko stanje. Na kraju prihvatnog ciklusa instrukcija je smeštena u instrukcijskom registru. Kada je instrukcijsko polje dekodirano upravljačka matrica automatski generiše pravu izvršnu rutinu. Po završetku izvršnog ciklusa kružni brojač se resetuje i počinje sledeći mašinski ciklus.

Obrada podataka se završava kada se HTL instrukcija učita u instrukcijski registar.s

---

10 – 8 Mikroprogramiranje

Upravljačka matrica sa slike 10-15 je jedan način za generisanje mikroinstrukcija koje su potrebne za svaki izvršni ciklus. Sa većim setom instrukcija upravljačka matrica postaje veoma komplikovana i zahteva stotine ili čak hiljade logičkih kola. Zbog toga je *hardverska kontrola* (logička kola povezana u matricu) primorala dizajnere da pronađu alternativni način za kreiranje upravljačkih reči koje pokreću računar.

Alternativa je *mikroprogramiranje*. Osnovna ideja je da se mikroinstrukcije smeste u ROM umesto da se kreiraju upravljačkom matricom. Ovaj pristup je uprostio problem projektovanja kontrolera – sekvencera.

Storing the Microprogram (Skladištenje mikroprograma)

Dodeljivanjem adresa i uključivanjem prihvatne rutine možemo doći do SAP–1 mikroinstrukcija prikazanih u tabeli 10-6. Ove mikroinstrukcije mogu biti skladištene u *upravljačkom ROM-u* sa prihatnom rutinom na adresi od 0H do 2H, LDA rutinom na adresi od 3H do 5H, ADD rutinom na adresi 6H do 8H, SUB rutinom na adresi 9H do BH i OUT rutinom na adresi od CH do EH.

Da bi se pristupilo bilo kojoj rutini moramo da pristupimo tačno određenim adresama. Na primer, da bismo dobili ADD rutinu moramo da pristupimo adresama, 6H, 7H i 8H. Da bismo pristupili OUT rutini moramo da pristupimo adresama CH, DH I EH. Dakle, pristupanje bilo kojoj rutini zahteva tri koraka:

1. Poznavanje početne adrese rutine
2. Prolazak kroz adrese rutine korak po korak
3. Učitavanje adresa u upravljački ROM

Adresni ROM

Slika 10-16 pokazuje kako programirati SAP–1 računar. Ima *adresni ROM*, *unapred postavljen brojač* i *upravljački ROM*. Adresni ROM sadrži početne adrese svake rutine iz tabele 10-6. Drugim rečima, adresni ROM sadrži podatke smeštene u tabeli 10-7. Kao što je prikazano, početna adresa LDA rutine je 0011, početna adresa ADD rutine je 0110 itd.

Kada bitovi operacionog koda upravljaju adresnim ROM-om početna adresa se generiše. Na primer, ako se izvršava ADD instrukcija $I_7I_6I_5I_4$ ima vrednost 0001. Ovo je ulazni signal u adresni ROM. Izlazni signal iz adresnog ROM-a je 0110.

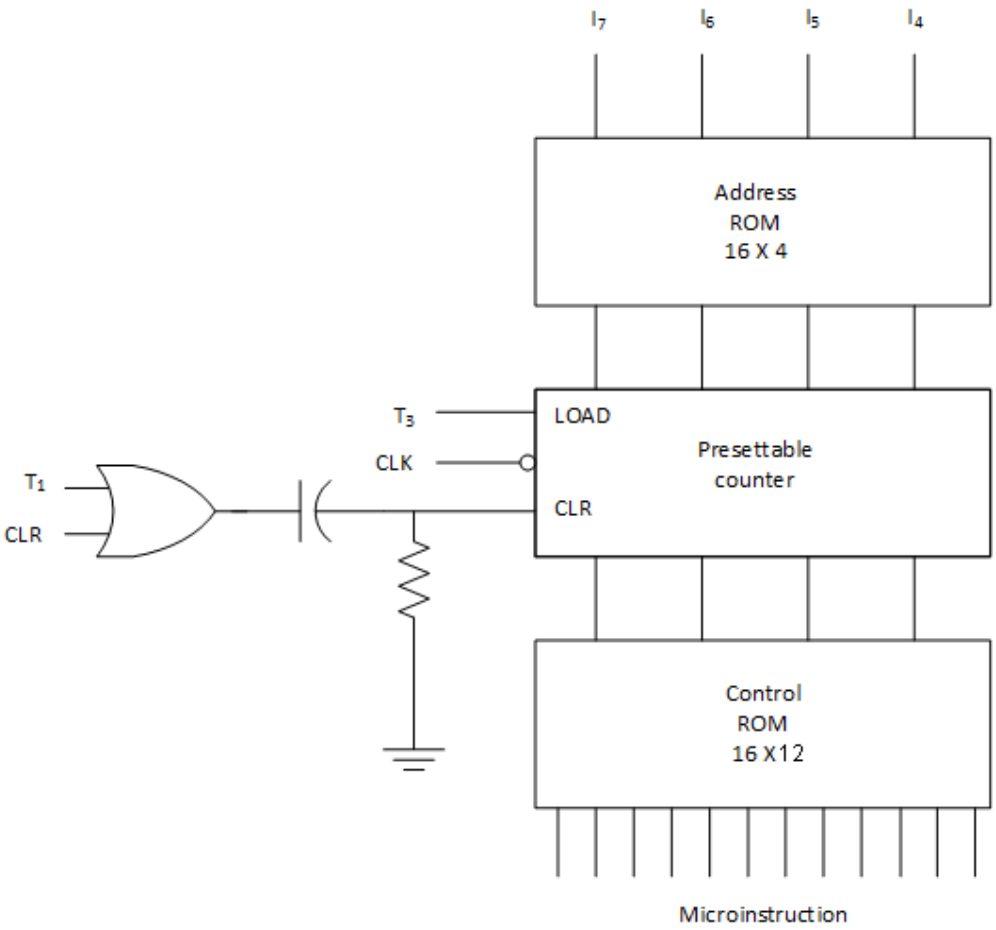
Tabela 10-6 – SAP-1 upravljački ROM

Adresa	Sadržaj	Rutina	Aktivni biti
0H	5E3H	Fetch	$E_P, \overline{L}_M$
1H	BE3H		C_P
2H	263H		$\overline{CE}, \overline{L_I}$
3H	1A3H	LDA	$\overline{L}_M, \overline{E}_I$
4H	2C3H		$\overline{CE}, \overline{L}_A$
5H	3E3H		Nijedan
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	$E_A, \overline{L}_O$
DH	3E3H		Nijedan
EH	3E3H		Nijedan
FH	X	X	Ne koristi se

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

Tabela 10-7 – Adresni ROM

Adresa	Sadržaj	Rutina
0000	0011	LDA
0001	0110	ADD
0010	1001	SUB
0011	XXXX	/
0100	XXXX	/
0101	XXXX	/
0110	XXXX	/
0111	XXXX	/
1000	XXXX	/
1001	XXXX	/
1010	XXXX	/
1011	XXXX	/
1100	XXXX	/
1101	XXXX	/
1110	1100	OUT
1111	XXXX	/



Slika 10-16 – Mikroprogramirano upravljanje SAP-1

Presettable Counter (Brojač sa upisom početnog stanja)

Kada je T3 na logičkoj jedinici, brojač učitava početnu adresu iz adresnog ROM-a. Tokom ostalih T stanja brojač broji, počev od datog učitanog stanja naviše.

Na početku, CLR signal na logičkoj jedinici iz clear – start debouncer-a se diferencira da bi se dobio uzak pozitivan signal. Ovo resetuje brojač. Kada računar krene sa radom izlaz iz brojača je 0000 tokom T_1 stanja, 0001 tokom T_2 i 0010 tokom T_3. Svaki prihvatni ciklus je isti jer 0000, 0001 i 0010 izlaze iz brojača tokom T_1, T_2 i T_3 stanja.

Operacioni kod u instrukcijskom registru kontroliše izvršni ciklus. Ako se prihvati ADD instrukcija $I_7I_6I_5I_4$ biti imaju vrednost 0001. Ovi biti operacionog koda upravljaju adresnim ROM-om, kreirajući 0110 na izlazu (Tabela 10-7). Ova početna adresa je ulaz za unapred postavljen brojač. Kada je T_3 na logičkoj jedinici sledećva negativna ivica CLK signala učitava 0110 u unapred postavljen brojač. Brojač je sada postavljen i brojanje može da se nastavi na početnoj adresi ADD rutine. Izlaz iz brojača je 0110 tokom T_4 stanja, 0111 toko T_5 stanja i 1000 tokom T_6 stanja.

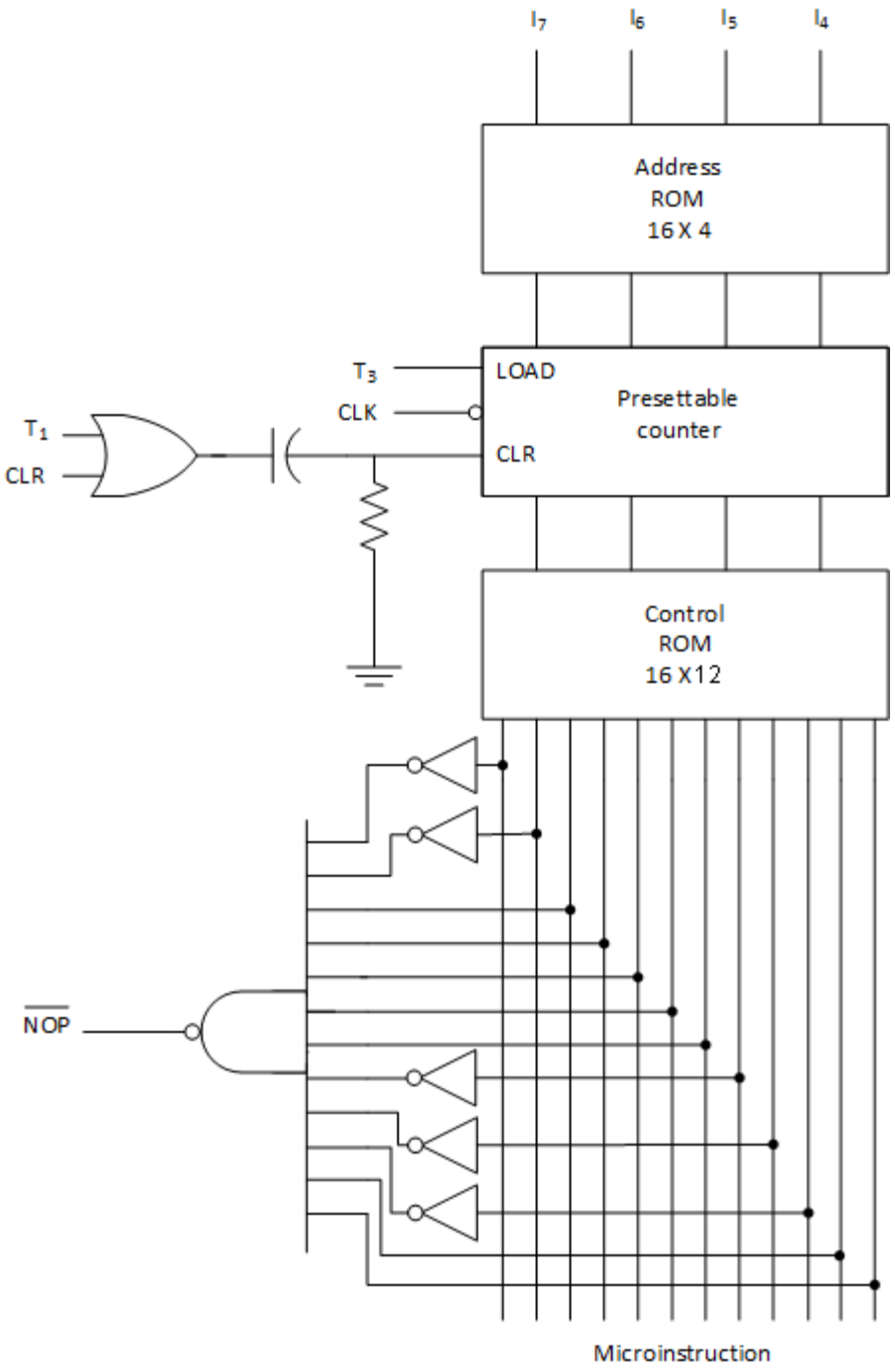
Kada T_1 stanje počne početna ivica T_1 signala se diferencira da bi se dobio uzak pozitivan signal koji resetuje brojač na 0000, početnu adresu prihvatne rutine, a zatim počinje nov mašinski ciklus.

Control ROM (Upravljački ROM)

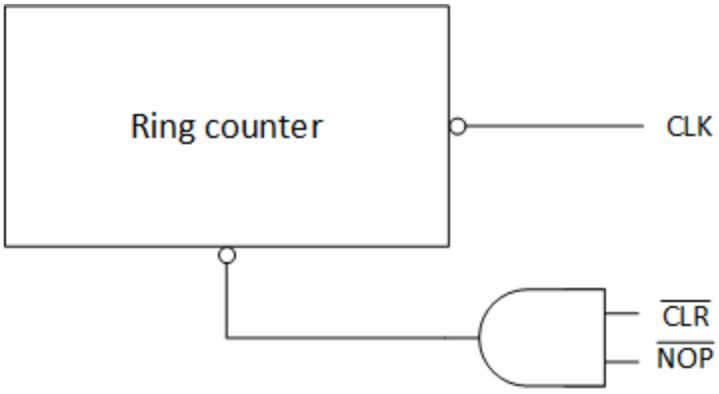
Upravljački ROM čuva SAP–1 mikroinstrukcije. Tokom prihvatnog ciklusa on prima adrese 0000, 0001 i 0010. Dakle, njegovi izlazi su 5E3H, BE3H, 263H.
Ove mikroinstrukcije, koje se nalaze u tabeli 10-6, kreiraju adresno, inkrementalno i memorijsko stanje.

Ako se izvršava ADD instrukcija upravljački ROM prima adrese 0110, 0111 i 1000 tokom izvršnog ciklusa. Njegovi izlazi su 1A3H, 2E1H i 3C7H.
Ove mikroinstrukcije izvršavaju operaciju sabiranja na način koji smo prethodno objasnili.

Za drugi primer, pretpostavimo da se izvršava OUT instrukcija. Tada je operacioni kod 1110, a početna adresa je 1100 (Tabela 10-7). Tokom izvršnog ciklusa izlaz iz brojača je 1100, 1101 i 1110. Izlaz iz upravljačkog ROMa je 3F2H, 3E3H, 3E3H (Tabela 10-6). Ova rutina prenosi sadržaj iz akumulatora do izlaznog porta.



Slika 10-17 – Promenljivi mašinski ciklus



Slika 10-18

Variable Machine Cycle (Promenljivi mašinski ciklus)

Mikroinstrukcija 3E3H u tabeli 10-6 je nop. Pojavljuje se jednom u LDA rutini i dvaput u OUT rutini. Ovi nop-ovi se koriste u SAP–1 da bi se dobio *fiksiran mašinski ciklus* za sve instrukcije. Drugim rečima, svaki mašinski ciklus traje tačno šest T stanja, bez obzira koja i kakva je instrukcija. U nekim računarima fiksiran mašinski ciklus je prednost. Ali kada je brzina važna nop-ovi su gubljenje vremena i mogu da se eliminišu.

Jedan način da se ubrza rad SAP–1 je da se preskoči svako T stanje koje sadrži nop. Redizajniranjem kola sa slike 10-16 možemo eliminisati nop stanja. Ovo će skratiti trajanje mašinskog ciklusa LDA instrukcije na pet stanja (T_1, T_2, T_3, T_4, T_5). Takođe skraćuje mašinski ciklus OUT instrukcije na četiri T stanja (T_1, T_2, T_3 I T_4).

Slika 10-17 prikazuje jedan način da se dobije *promenljivi mašinski ciklus*. Kod LDA instrukcije, izvršavanje je potpuno isto kao i ranije od T_1 do T_5 stanja. Kada počne T_6 stanje upravljački ROM kreira 3E3H na izlazu (nop mikroinstrukcija). NI logičko kolo detektuje ovaj nop istog trenutka i daje $\overline{NOP}$ signal na logičkoj nuli. $\overline{NOP}$ se vraća u kružni brojač preko I logičkog kola, kao što se vidi na slici 10-18. Ovo resetuje kružni brojač na T_1 stanje i počinje novi mašinski ciklus. Ovo skraćuje trajanje mašinskog ciklusa LDA instrukcije sa šest na pet stanja.

Kod OUT instrukcije prvi nop se pojavi u stanju T_5. U ovom slučaju, čim stanje T_5 počne upravljački ROM kreira 3E3H na izlazu što se detektuje NI kolom. $\overline{NOP}$ signal na logičkoj nuli tada resetuje kružni brojač na T_1 stanje. Na ovaj način, skratili smo trajanje mašinskog ciklusa OUT instrukcije sa šest na četiri stanja.