Centralenheten: ALU, dataväg och minne

Dagens föreläsning:

Kompendium kapitel 7
Arbetsbokens kapitel 11,12

Ur innehållet:

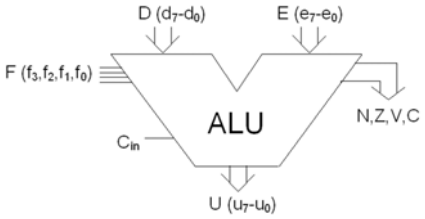
- ❑ RTN, beskrivande notation
- ❑ Dataväg med Aritmetik/Logik- enhet (ALU)
- ❑ Primärminnet, läsning och skrivning
- ❑ En manuell styrenhet

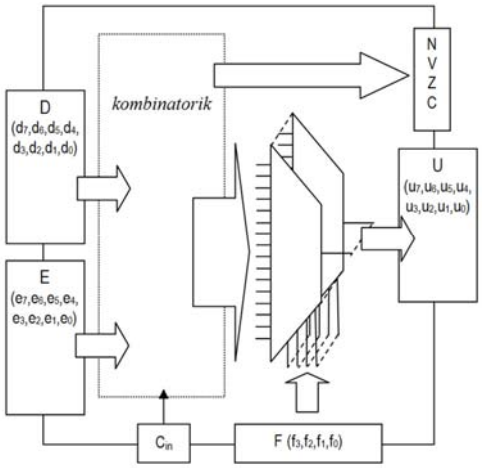
RTN - Register Transfer Notation

- Förenklat skrivsätt för att specificera operationer där register ingår
- I skrivsättet används enkla symboler och för att speciellt ange att RTN-notation avses använder vi ett avvikande typsnitt.

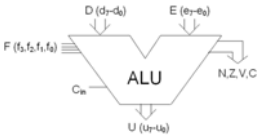
operator	operation
n	Konstant uttryckt i talbas 10
N_r	Konstanten N uttryckt i talbasen r.
→	Kopiering
+	Addition
-	Subtraktion
∧	Logiskt "OCH" (AND)
∨	Logiskt "ELLER" (OR)
⊕	Logiskt "EXKLUSIVT ELLER" (XOR)
Opr<<1 (*d*)	"Opr" skiftas vänster ett steg. Biten *d* skiftas in i den minst signifikanta positionen.
(*d*) 1>>Opr	"Opr" skiftas höger. Biten *d* skiftas in i den mest signifikanta positionen.
Opr'	Bitvis komplementering av operanden "Opr"
=	likhet
!=	olikhet
>	större än
<	mindre än
>=	större än eller likhet
<=	mindre än eller likhet

Aritmetik- logikenhet (ALU)





ALU's funktioner



funktion				*operation*	*utsignaler*											
f_3	f_2	f_1	f_0	RTN	u_7	u_6	u_5	u_4	u_3	u_2	u_1	u_0	N	Z	V	C
0	0	0	0	U= 0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	1	U= FD_{16}	1	1	1	1	1	1	0	1	1	0	0	0
0	0	1	0	U= FE_{16}	1	1	1	1	1	1	1	0	1	0	0	0
0	0	1	1	U= FF_{16}	1	1	1	1	1	1	1	1	1	0	0	0
0	1	0	0	U= E	e_7	e_6	e_5	e_4	e_3	e_2	e_1	e_0	u_7	(1)	0	0
0	1	0	1	U= $D_{1k}+C_{in}$									u_7	(1)	(8)	(7)
0	1	1	0	U= DvE	$d_7 \vee e_7$	$d_6 \vee e_6$	$d_5 \vee e_5$	$d_4 \vee e_4$	$d_3 \vee e_3$	$d_2 \vee e_2$	$d_1 \vee e_1$	$d_0 \vee e_0$	u_7	(1)	0	0
0	1	1	1	U= D∧E	$d_7 \wedge e_7$	$d_6 \wedge e_6$	$d_5 \wedge e_5$	$d_4 \wedge e_4$	$d_3 \wedge e_3$	$d_2 \wedge e_2$	$d_1 \wedge e_1$	$d_0 \wedge e_0$	u_7	(1)	0	0
1	0	0	0	U= D⊕E	$d_7 \oplus e_7$	$d_6 \oplus e_6$	$d_5 \oplus e_5$	$d_4 \oplus e_4$	$d_3 \oplus e_3$	$d_2 \oplus e_2$	$d_1 \oplus e_1$	$d_0 \oplus e_0$	u_7	(1)	0	0
1	0	0	1	U= D+C_{in}									u_7	(1)	(2)	(3)
1	0	1	0	U= D+FF_{16}									u_7	(1)	(2)	(3)
1	0	1	1	U= D+E+C_{in}									u_7	(1)	(2)	(3)
1	1	0	0	U= D+$E_{1k}+C_{in}$									u_7	(1)	(2)	(3)
1	1	0	1	U= D<<1 (C_{in})	d_6	d_5	d_4	d_3	d_2	d_1	d_0	C_{in}	u_7	(1)	0	(4)
1	1	1	0	U= (C_{in}) 1>>D	C_{in}	d_7	d_6	d_5	d_4	d_3	d_2	d_1	u_7	(1)	(6)	(5)
1	1	1	1	U= (d_7)1>>D	d_7	d_7	d_6	d_5	d_4	d_3	d_2	d_1	u_7	(1)	0	(5)

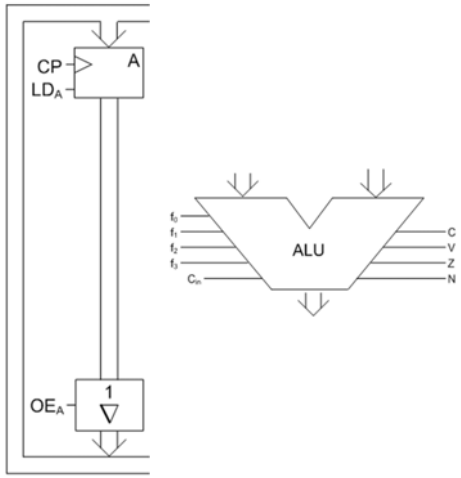
(1) $Z = \overline{u_7} \wedge \overline{u_6} \wedge \overline{u_5} \wedge \overline{u_4} \wedge \overline{u_3} \wedge \overline{u_2} \wedge \overline{u_1} \wedge \overline{u_0}$, dvs. Z=1 då samtliga bitar i register U är 0, Z=0 annars.

(2) $V = (\overline{u_7} \wedge d_7 \wedge e_7) \vee (u_7 \wedge \overline{d_7} \wedge \overline{e_7})$, dvs. V-flaggan sätts enligt reglerna för tvåkomplementsaritmetik.

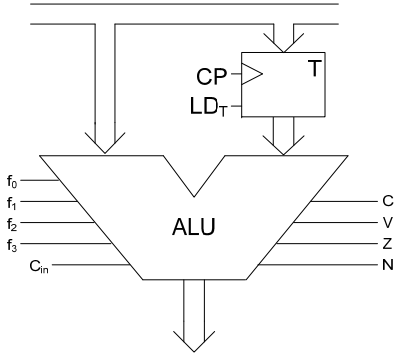
(3) C = c_8, dvs. carry ut från additionen av de mest signifikanta siffrorna.
(4) C = utskiftad bit, dvs. bit d_7 före vänsterskiftet.
(5) C = utskiftad bit, dvs. bit d_0 före högerskiftet.
(6) V = $C_{in} \oplus d_7$ före skift, dvs. sätts till 1 om skiftet föranleder teckenbyte.
(7) C ettställs om D=0, C nollställs annars.
(8) V ettställs om D=$(10000000)_2$, V nollställs annars.

Anslutning av ALU till datavägen

Hur kopplar vi in en ALU till datavägen?

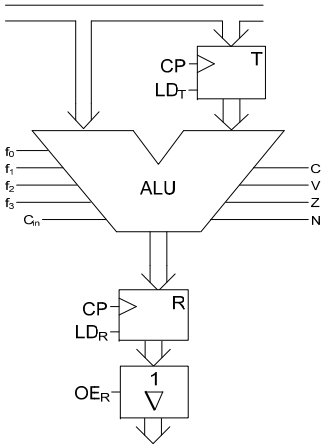


Temporärregister (T) för lagring av indata



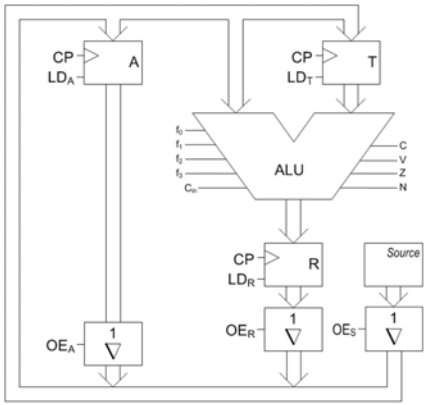
Bussen kan bara innehålla en av ALU'ns operander åt gången

Resultatregister (R) för lagring av utdata

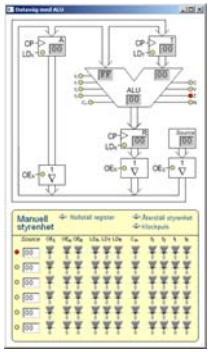


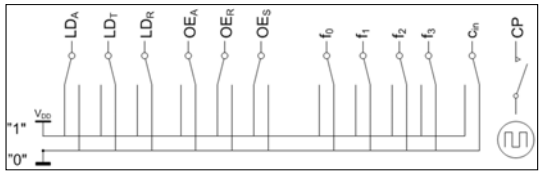
Eftersom resultatet i ALU'ns U-register ändras direkt om någon av ingångarna ändras måste det vara möjligt att spara värdet i ett register.

Dataväg med ALU



funktion				operation
f_3	f_2	f_1	f_0	
0	0	0	0	RTN
0	0	0	1	U= 0
0	0	1	0	U= FD_{16}
0	0	1	1	U= FE_{16}
0	1	0	0	U= FF_{16}
0	1	0	1	U= E
0	1	1	0	U= $D_{16}+C_{in}$
0	1	1	1	U= D∨E
1	0	0	0	U= D∧E
1	0	0	1	U= D⊕E
1	0	1	0	U= D+C_{in}
1	0	1	1	U= D+FF_{16}
1	1	0	0	U= D+E+C_{in}
1	1	0	1	U= D<<1 (C_{in})
1	1	1	0	U= (C_{in}) 1>>D
1	1	1	1	U= (d_7)1>>D

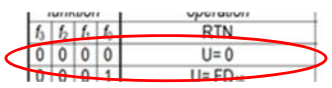


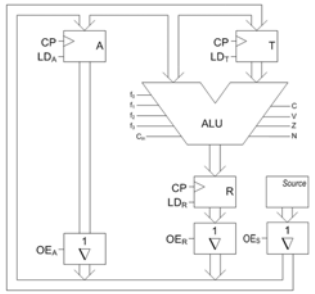


Manöverpanel – strömställare för styrsignaler.

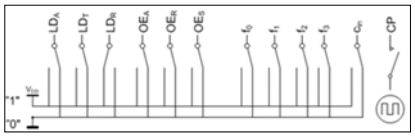
Exempel: Nollställning av register A, 0→A

Cykel	Operation (RTN)	Aktiva styrsignaler	Beskrivning
1	0→R	LD_R	ALU'ns U-register nollställs ty F(0), dvs $f_3=f_2=f_1=f_0=0$. Vid klockpulsen överförs U till R.
2	R→A	OE_R, LD_A	Innehållet i register R överförs till register A

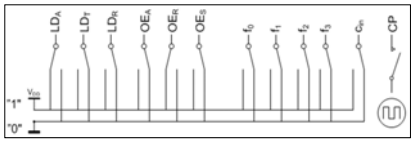




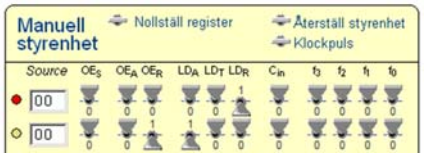
1



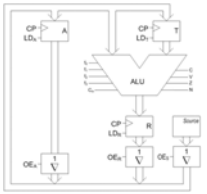
2



I DigiFlisp:



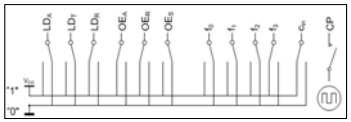
Exempel: Inkrementering av registerinnehåll A, A+1→A



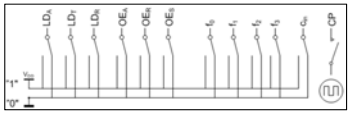
funktion				operation
f_3	f_2	f_1	f_0	
0	0	0	0	RTN
0	0	0	1	U= 0
0	0	1	0	U= FD$_{16}$
0	0	1	1	U= FE$_{16}$
0	1	0	0	U= FF$_{16}$
0	1	0	1	U= E
0	1	1	0	U= D+C$_{in}$
0	1	1	1	U= DvE
1	0	0	0	U= D∧E
1	0	0	1	U= D⊕E
1	0	1	0	U= D+C$_{in}$
1	0	1	1	U= D+FF$_{16}$
1	1	0	0	U= D+E+C$_{in}$
1	1	0	1	U= D+E$_{16}$+C$_{in}$
1	1	1	0	U= D<<1 (C$_{in}$)
1	1	1	1	U= (C$_{in}$) 1>>D
				U= (d$_7$)1>>D

Cykel	Operation (RTN)	Styrsignaler	Beskrivning
1	A+1→R	OE_A; C_{in}; f_3; LD_R	A kopplas till ALU'n Carry in ingår i operationen F(9), dvs $f_3=1, f_2=f_1=0, f_0=1$. Vid klockpulsen överförs U till R.
2	R→A	OE_R; LD_A	Innehållet i register R överförs till register A

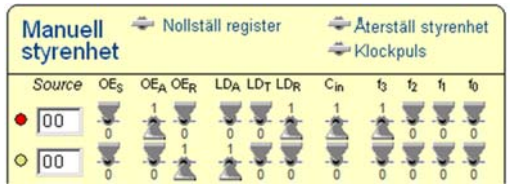
1



2

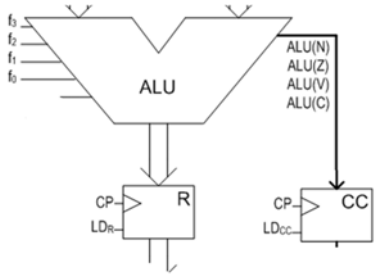


I DigiFlisp:



Flaggregister, "Condition Codes" (CC)

- ALU:ns sätt att påverka flaggorna är "flyktigt", dvs i samma ögonblick som styrsignalerna F ändras så ändras också vanligtvis någon eller några av flaggorna N,V,Z, och C.
- Flaggorna från en ALU-operation måste kunna sparas och det blir därför nödvändigt att införa ett register (CC) för detta.



Det är bitarna i CC vi menar (med N,Z,V och C) om inget annat sägs uttryckligt.

Flaggregister, olika sätt att påverka

- Olika instruktioner påverkar CC på olika sätt
- Flaggor måste kunna påverkas individuellt

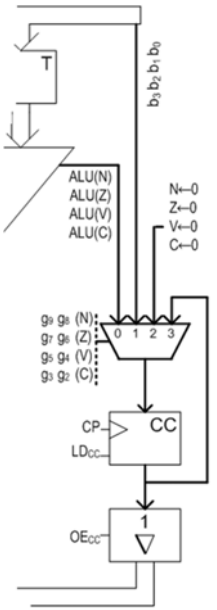
Logiken (en väljare) utformas så att varje flagga i CC, oberoende av övriga, kan väljas från en av fyra källor:

ALU :
Flaggan i CC kopieras från ALU:n dvs. sätts baserat på resultatet av ALU:ns senaste operation

Buss:
Flaggan i CC kopieras från datavägens buss

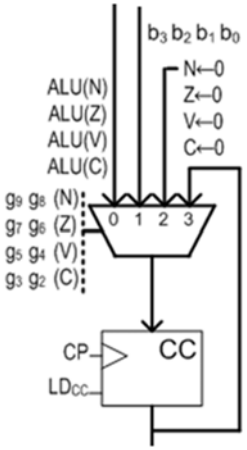
Konstant:
Flaggor N,Z,V,C nollställs

Ingen ändring:
Aktuella värden återförs (ändras ej).



Flaggregister, väljarfunktioner

8 olika väljarsignaler (g_2-g_9) kan användas för att styra flaggsättningen i CC:



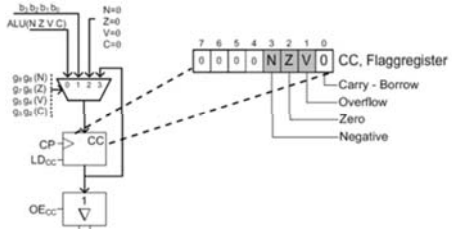
g_3	g_2	C väljs enligt:	RTN	g_5	g_4	V väljs enligt:	RTN
0	0	C tas från ALU'n	ALU(C) → C	0	0	V tas från ALU'n	ALU(V) → V
0	1	C tas från bit 0 på bussen	b_0 → C	0	1	V tas från bit 1 på bussen	b_1 → C
1	0	Återställning (nollställning) av C	0 → C	1	0	Återställning (nollställning) av V	0 → V
1	1	C återförs (ändras ej)		1	1	V återförs (ändras ej)	

g_7	g_6	Z väljs enligt:	RTN	g_9	g_8	N väljs enligt:	RTN
0	0	Z tas från ALU'n	ALU(Z) → Z	0	0	N tas från ALU'n	ALU(N) → N
0	1	Z tas från bit 2 på bussen	b_2 → Z	0	1	N tas från bit 3 på bussen	b_3 → N
1	0	Återställning (nollställning) av Z	0 → Z	1	0	Återställning (nollställning) av N	0 → N
1	1	Z återförs (ändras ej)		1	1	N återförs (ändras ej)	

Exempel: Nollställ C-flaggan, 0→C

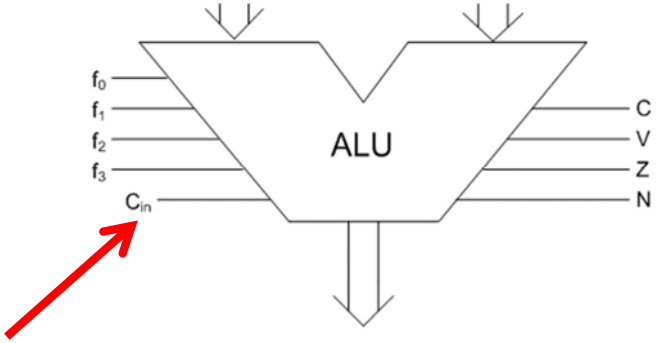
- Operationen ska nollställa C-flaggan i CC medan övriga flaggor ska förbli oförändrade. Bitarna b_4 t.o.m b_7 i CC används inte och kan heller inte påverkas.
- I detta fall utnyttjar vi lämpligen att '0' kan kopplas direkt till flaggorna via väljaren, dvs. C tas från '0' medan övriga flaggor återkopplas.

Cykel	Operation (RTN)	Aktiva styrsignaler	Beskrivning
1	0→C	g_3; g_5; g_4; g_7; g_6; g_9; g_8; LD_{CC}	$G_C(2)$, C tas från '0' $G_V(3)$, V återkopplas $G_Z(3)$, Z återkopplas $G_N(3)$, N återkopplas Vid klockpuls laddas CC



ALU, kontroll av C_{in}-funktion

- C_{in}-funktion bestäms av operation (F), normalt 0 eller 1.
- Operationer som "upprepad addition" och "upprepad subtraktion" kräver dock att vi kan välja C eller C' som C_{in}.

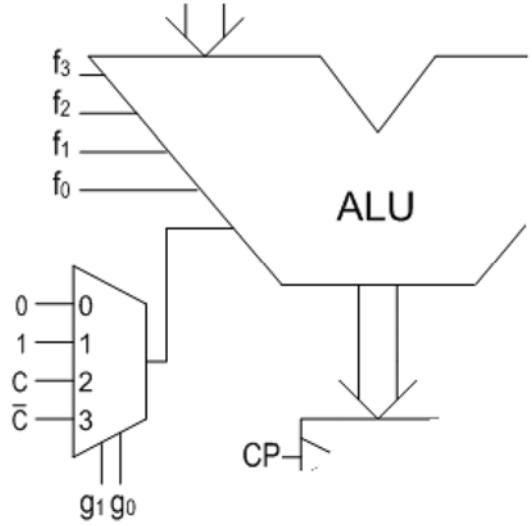


funktion				operation
f_3	f_2	f_1	f_0	RTN
0	0	0	0	U= 0
0	0	0	1	U= FD_{16}
0	0	1	0	U= FE_{16}
0	0	1	1	U= FF_{16}
0	1	0	0	U= E
0	1	0	1	U= D_{15}+C_{in}
0	1	1	0	U= D∨E
0	1	1	1	U= D∧E
1	0	0	0	U= D⊕E
1	0	0	1	U= D+C_{in}
1	0	1	0	U= D+FF_{16}
1	0	1	1	U= D+E+C_{in}
1	1	0	0	U= D+E_{15}+C_{in}
1	1	0	1	U= D<<1 (C_{in})
1	1	1	0	U= (C_{in}) 1>>D
1	1	1	1	U= (d_7)1>>D

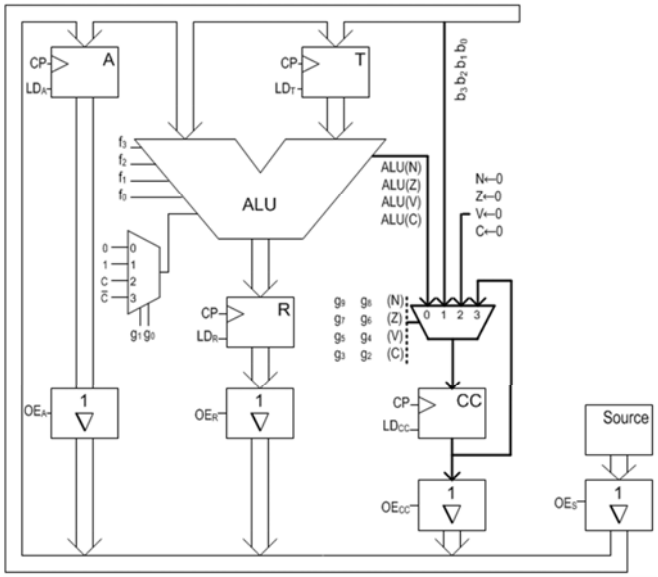
Väljare för kontroll av c_{in}-funktion

- Vi kopplar en "1 av 4-väljare" till ingången C_{in}
- Inför väljarsignaler g1, g0 för att styra funktionen

g_1	g_0	Signal till C_{in}	RTN
0	0	konstant värde 0	C_{in} =0
0	1	konstant värde 1	C_{in} =1
1	0	C-flagga från CC	C_{in} =C
1	1	C-flagga invers från CC	C_{in} =C'



Dataväg med flaggregister (CC) och valbar c_{in}-funktion



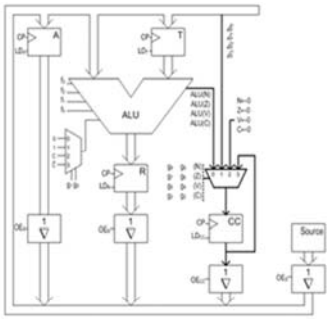
Exempel: Omedelbar adressering, 80_{16} →A

Skapa en styrsignalsekvens som placerar konstanten 80_{16} i register A. Flaggorna i CC ska påverkas så att N och Z sätts baserat på operanden (dvs. 80_{16}), V och C ska nollställas.

Lösning:
RTN för operationen blir:

$80_{16} \rightarrow A$; ALU(N) → N; ALU(Z) → Z; 0→ V; 0→ C;

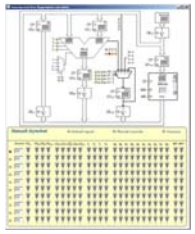
funktion				operation	flaggor			
f_3	f_2	f_1	f_0	RTN	N	Z	V	C
0	0	0	0	U=0	0	1	0	0
0	0	0	1	U=FD$_{16}$	1	0	0	0
0	0	1	0	U=FE$_{16}$	1	0	0	0
0	0	1	1	U=FF$_{16}$	1	0	0	0
0	1	0	0	U=E	uz	(1)	0	0
0	1	0	1	U=D$_{16}$ + C_{in}	uz	(1)	(8)	(7)
0	1	1	0	U=D∨E	uz	(1)	0	0
0	1	1	1	U=D∧E	uz	(1)	0	0
1	0	0	0	U=D⊕E	uz	(1)	0	0
1	0	0	1	U=D + C_{in}	uz	(1)	(2)	(3)
1	0	1	0	U=D + FF$_{16}$	uz	(1)	(2)	(3)
1	0	1	1	U=D + E + C_{in}	uz	(1)	(2)	(3)
1	1	0	0	U=D + E$_{16}$ + C_{in}	uz	(1)	(2)	(3)
1	1	0	1	U=D<<1 (C_{in})	uz	(1)	(8)	(4)
1	1	1	0	U=(C_{in}) D >> 1	uz	(1)	(8)	(5)
1	1	1	1	U=(d_7) D >> 1	uz	(1)	0	(5)

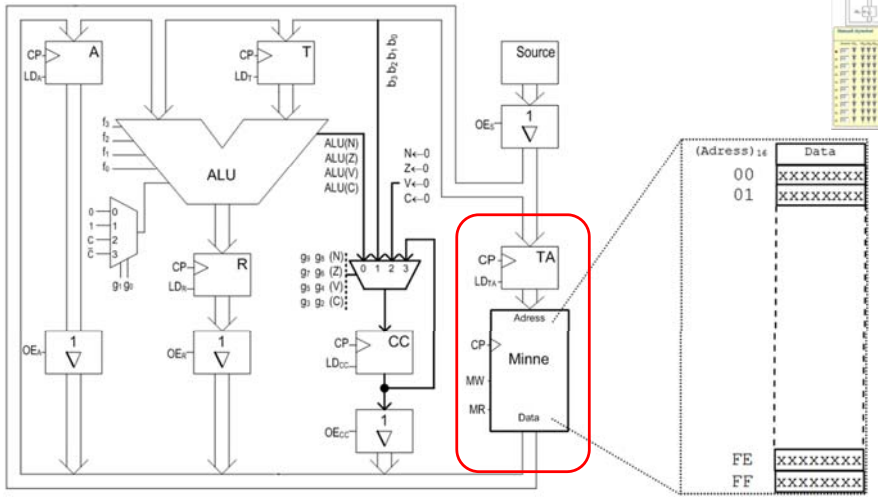


Klock-puls	Source	Output Enable				Load Enable					ALU(f)				MUX(g)								
		S	A	R	CC	A	T	TA	R	CC	3	2	1	0	9	8	7	6	5	4	3	2	1	0
1	80_{16}	1				1				1	1			1					1		1			

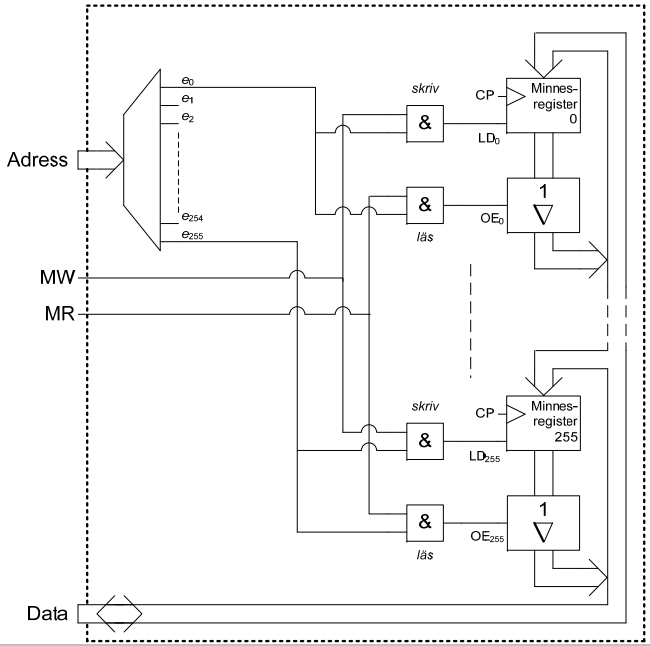
Vi ansluter minne

I DigiFlisp:





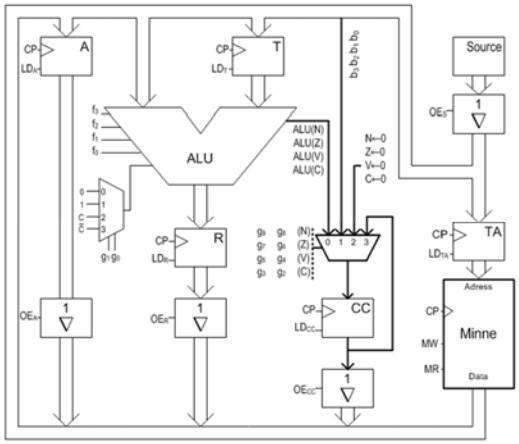
Princip Läs-/ Skriv-minne



Läs-cykel

Exempel:
Kopiera innehållet från adress 10_{16} i minnet till register A.

1. Placera adressen (10_{16}) i register TA:
   $10_{16} \rightarrow$TA
2. Överför data från minne till register:
   M(TA) → A

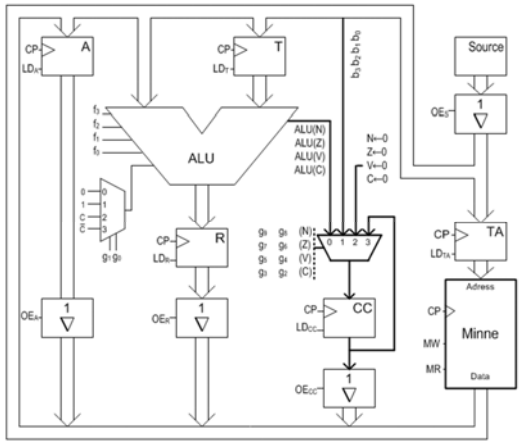


RTN	steg	Source	OE_S	OE_A	OE_R	OE_{CC}	LD_A	LD_T	LD_{TA}	LD_R	LD_{CC}	f_3	f_2	f_1	f_0	g_9	g_8	g_7	g_6	g_5	g_4	g_3	g_2	g_1	g_0	MR	MW
$10_{16} \rightarrow$ TA	1	10	1						1																		
M(TA)→ A	2						1																			1	

Skriv-cykel

Exempel:
Kopiera innehållet från register A till adress 11_{16} i minnet.

1. Placera adressen (11_{16}) i register TA:
   $11_{16} \rightarrow$TA
2. Överför data från register till minne:
   A → M(TA)



RTN	steg	Source	OE_S	OE_A	OE_R	OE_{CC}	LD_A	LD_T	LD_{TA}	LD_R	LD_{CC}	f_3	f_2	f_1	f_0	g_9	g_8	g_7	g_6	g_5	g_4	g_3	g_2	g_1	g_0	MR	MW
$11_{16} \rightarrow$ TA	1	11	1						1																		
A → M(TA)	2			1																							1

Exempel: Addition, A+M(10_{16})→A

Utför addition av innehållet i register A med innehållet på adress 10_{16} i minnet.

Dela upp RTN-beskrivningen i delar som kan åstadkommas inom en klockcykel, vi har två operander, placera den andra i register T

1. Placera adressen (10_{16}) i register TA:
   $10_{16} \rightarrow$TA
2. Överför data från minne till register T:
   M(TA) →T

Nu är vi klara att utföra operationen:

3. A+T →R; ALU(N,Z,V,C) →CC

Resultatet finns nu i R, vi behöver bara återföra det till A:

4. R→A

Vid addition ska flaggorna påverkas så att:

- N: Ettställs om resultatets teckenbit (b_7) får värdet 1.
- Z: Ettställs om samtliga åtta bitar i resultatet blir noll.
- V: Ettställs om 2-komplementspill uppstår.
- C: Ettställs om summan vid additionen ej ryms i åtta bitar.

i övriga fall nollställs respektive flaggbit.

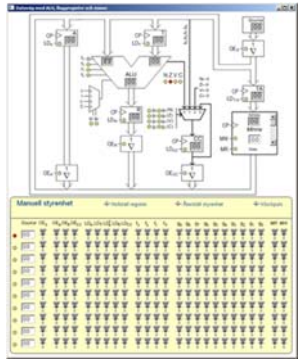
ALU:ns flaggsättning vid addition:

- N: u_7.
- Z: Ettställs om samtliga åtta bitar i U blir noll, nollställs annars
- V: Sätts enligt reglerna för tvåkomplementsaritmetik.
- C: c_8, dvs. carry ut från additionen av de mest signifikanta siffrorna.

Forts. exempel: Addition, A+M(10_{16})→A

Från RTN-beskrivningen fyller vi i styrsignalsekvensen

För att förenkla test, använder vi den första raden för att placera något lämpligt testvärde i register A



RTN	steg	Source	OE_S	OE_A	OE_R	OE_{CC}	LD_A	LD_T	LD_{TA}	LD_R	LD_{CC}	f_3	f_2	f_1	f_0	g_9	g_8	g_7	g_6	g_5	g_4	g_3	g_2	g_1	g_0	MR	MW
xx →A	1	xx	1				1																				
10_{16}→TA	2	10	1						1																		
M(TA) →T	3							1																		1	
A+T→R;ALU(N,Z,V,C)→CC	4			1						1	1	1		1	1												
R→A	5				1		1																				