Ext-14

FLEX-processorns styrenhet med fast logik

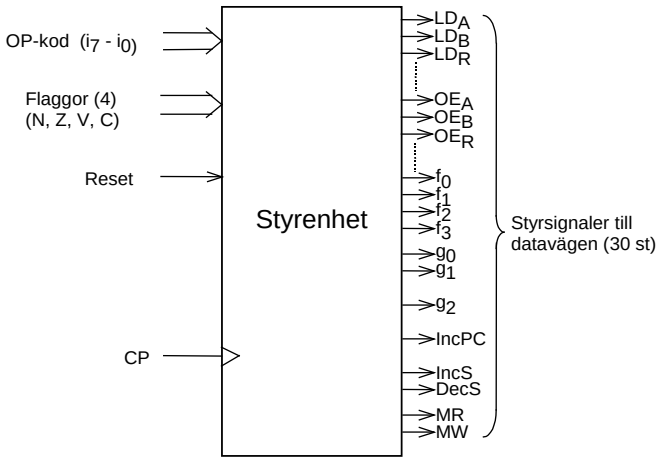
Ext-14

FLEX-processorns styrenhet med fast logik

En styrenhet för FLEX-processorn skall kunna generera alla styrsignaler till datavägen och minnet för samtliga instruktioner i instruktionslistan. Under körning av ett program måste därför ett stort antal olika styrsignalsekvenser av varierande längd och komplexitet genereras.

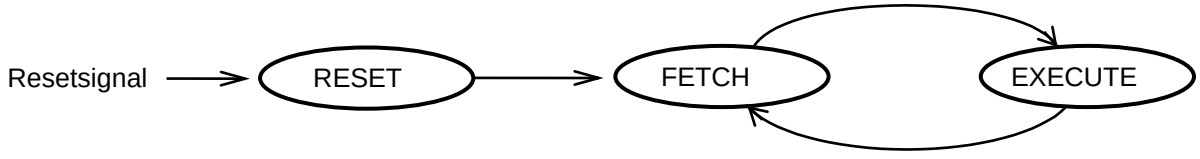
Vi har tidigare sett hur datavägen och minnet kan styras genom att styrsignalerna först ges rätta värden varpå en klockpuls CP (positiv flank) verkställer laddning av ett eller flera utvalda register. Efter klockpulsflanken byter man till nya styrsignalvärden och verkställer nästa laddning med nästa klockpulsflank osv. Byte av styrsignaler och laddning av register sker alltså i samma takt. Det är därför lämpligt att använda samma klockpulssignal CP till datavägen, minnet och styrenheten. Styrenheten kommer därför att utgöras av ett synkront sekvensnät med CP som klocksignal och samtliga styrsignaler till datavägen och minnet som utsignaler.

Som insignaler till styrenheten används innehållet i instruktionsregistret (dvs OP-koden) och innehållet i flaggregistret (dvs flaggorna). Styrenheten behöver OP-koden eftersom den bestämmer utförandefasen (EXECUTE) av instruktionen. Flaggorna behövs när en villkorlig instruktion skall utföras. För att processorn skall kunna startas eller återstartas behövs även en insignal (startsignal) Reset. Figur 1 visar styrenheten utifrån sett.



Figur 1. Styrenhet till FLEX-processorn.

En lämplig utgångspunkt för att bestämma styrenhetens interna uppbyggnad är den grova flödesplanen nedan. Den visar hur växling sker mellan interna tillstånd när instruktioner exekveras. Först konstateras att processorn måste starta från ett starttillstånd och utföra en startsekvens (RESET). Därefter skall den övergå till hämtfasen (FETCH), som följs av utförandefasen (EXECUTE) för den instruktion vars OP-kod finns i instruktionsregistret. Under normalt arbete skall den sedan ständigt växla mellan hämtfasen (FETCH) och utförandefasen (EXECUTE) enligt grafen.



Tillståndsgraf för styrenhet

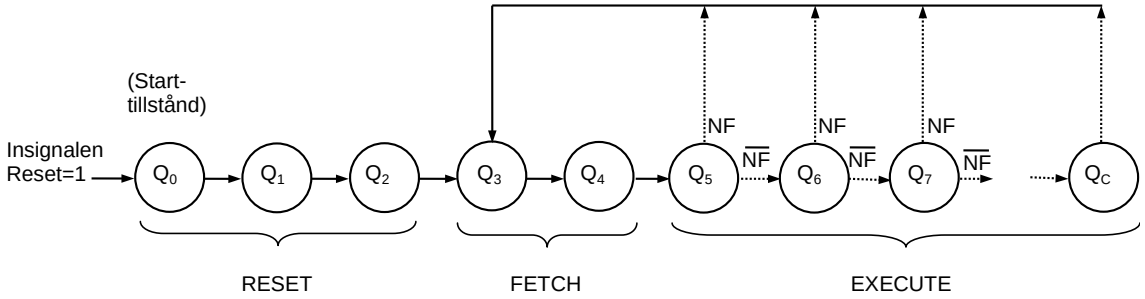
I häftet Ext-8 visas att RESET-sekvensen kräver tre tillstånd och FETCH-sekvensen två tillstånd.

Varje instruktion har ett unikt beteende i sin EXECUTE-sekvens, som därmed blir olika lång. Sekvensens längd bestäms ju av det nyttiga arbete den aktuella instruktionen skall utföra.

Vi antar till en början att det räcker med åtta tillstånd för att utföra allt som behöver göras i EXECUTE-fasen för den längsta instruktionen.

Maximala antalet tillstånd som styrenheten behöver genomlöpa är i så fall 3 + 2 + 8 = 13.

Figur 2 visar RESET-, FETCH- och EXECUTE-sekvensen. De 13 tillstånden har döpts Q_0 – Q_C.



Figur 2. Tillståndsgraf för FLEX-processorns styrenhet.

Av figur 2 framgår också att en ny styrsignal NF (Next FETCH) har införts, detta för att förenkla övergången från det sista tillståndet i en EXECUTE-sekvens till det första tillståndet Q_3 i FETCH-sekvensen. NF används enbart internt inom styrenheten. I det sista tillståndet i en EXECUTE-sekvens skall styrsignalen NF ges värdet 1.

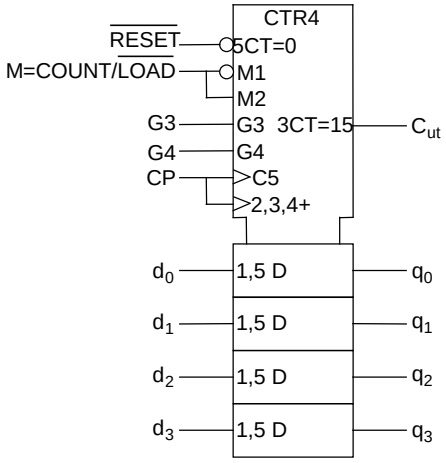
I varje tillstånd aktiveras ett antal av de sammanlagt (30 + 1) st styrsignalerna. Varje styrsignal är en Boolesk funktion av tillståndsvariablerna som definierar tillstånden (ringarna) och OP-koden. Eftersom det finns villkorliga instruktioner ingår även flaggorna i de Booleska uttrycken för vissa av styrsignalerna.

I tillståndsgrafen skall egentligen också finnas övergångar (pilar) från varje ring till ringen längst till vänster, dvs starttillståndet. Övergång till starttillståndet skall ske så fort styrenhetens insignal Reset = 1 vid positiv klockpulsflank. Detta är den enda påverkan insignalen Reset har på styrenheten.

Realisering av styrenhet med fast logik

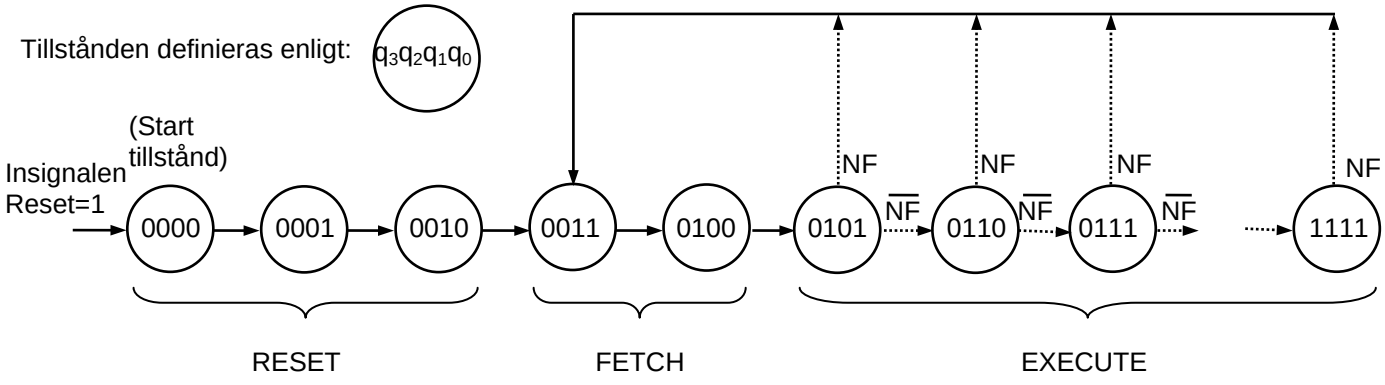
Det framgår av tillståndsgrafen i figur 2 att styrenheten liknar en räknare. Efter återställning, med insignalen Reset = 1, startar den från tillstånd Q_0 och räknar sedan uppåt genom RESET, FETCH- och EXECUTE-sekvensen. När det sista tillståndet i EXECUTE-sekvensen för den aktuella instruktionen har nåtts startar den om från första tillståndet i FETCH (Q_3).

Den laddningsbara 4-bitars räknaren 74HC163, som användes i arbetshäfte nr 2, passar utmärkt för realisering av styrenheten. Räknarens symbol visas i figur 3.



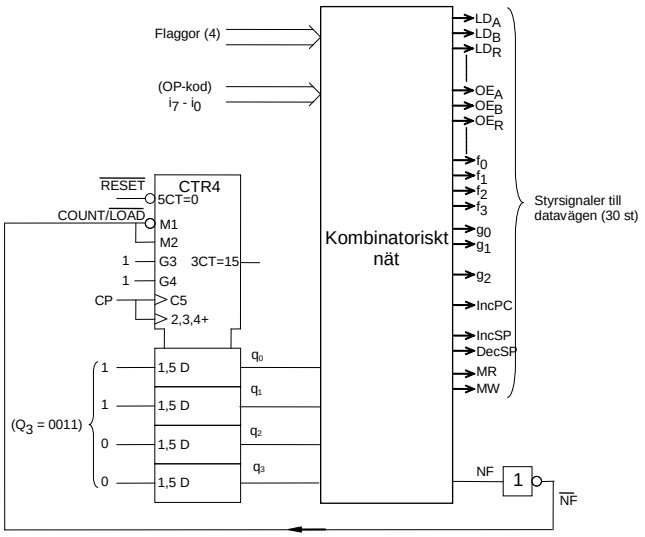
Figur 3. 4-bitars räknaren 74HC163.

Om man kodar tillstånden Q_i i tillståndsgrafen i figur 2 som $q_3q_2q_1q_0$ och väljer Q_0 – Q_C som 0000 -1100, får man samma tillståndssekvens som i räknaren. Tillståndsgrafen visas i figur 4, där EXECUTE-sekvensen dessutom har kompletterats med de tre extra tillstånden Q_D – Q_F, som man får på köpet genom att använda räknarens samtliga tillstånd.



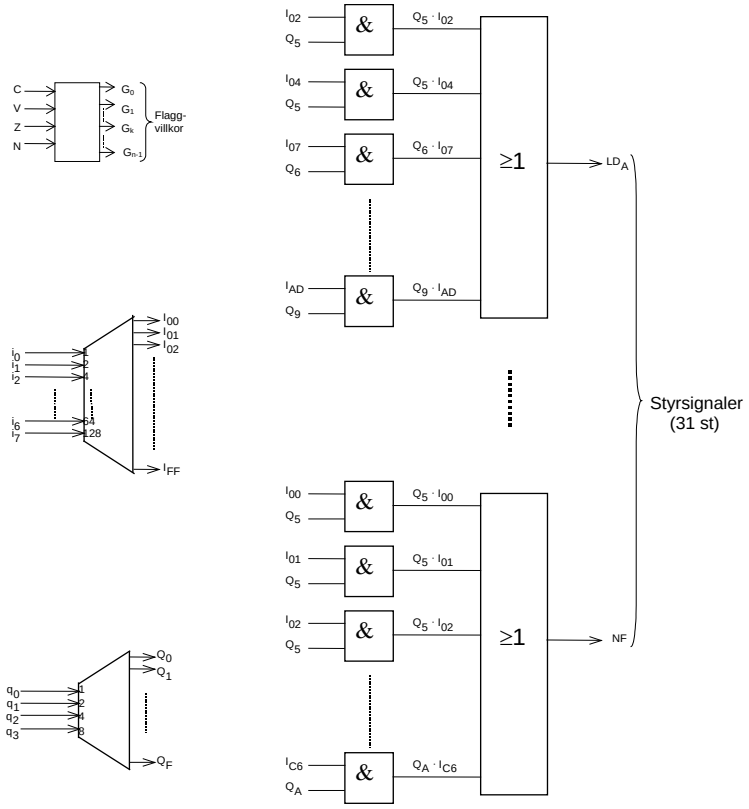
Figur 4. Exempel på tillståndskodning för FLEX-processorns automatiska styrenhet.

Kretsrealiseringen för denna tillståndsgraf blir mycket enkel och visas i figur 5.



Figur 5. Enkel realisering av FLEX-processorns automatiska styrenhet.

Det kombinatoriska nätet i figur 5 är uppbyggt enligt principen i figur 6. Förutom avkodning av tillståndsräknare och instruktionernas OP-koder innehåller det en s k AND-OR area där styrsignalerna skapas som (Booleska) summor av produkttermer. Produkttermerna består av tillståndsvariablerna q_0 - q_3, OP-kodbitarna i_0 - i_7 och eventuellt flaggbitarna om en villkorlig instruktion skall utföras.



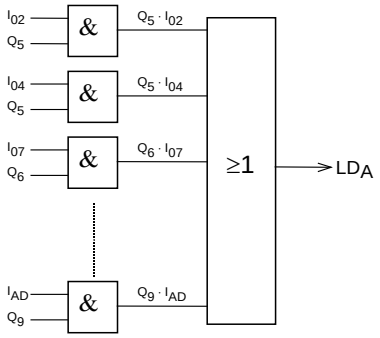
Figur 6 Princip för kombinatorisk del av styrenhet med fast kopplad logik.

Av figur 6 framgår att räknarens utsignaler q_3, q_2, q_1 och q_0 är kopplade till en 4/16-avkodare med de sexton utsignalerna Q_0 – Q_F. Detta innebär t ex att när räknarens tillstånd $q_3q_2q_1q_0$ = $(0011)_2$ = 3, är Q_3 = 1, medan övriga Q-värden från avkodaren är 0. Det framgår också att instruktionsregistrets åtta bitar i_0 - i_7 är kopplade till en 8/256-avkodare med de 256 utsignalerna I_{00} -I_{FF}.

Genom att studera hur styrsignalen LD_A bildas i det kombinatoriska nätet i figur 6 får man en god uppfattning om hur hela nätet är konstruerat.

Styrsignalen LD_A, är utsignal från en OR-grind med ett stort antal ingångar enligt figur 7 nedan. Den är alltså A-registrets "load signal" och måste vara aktiv (=1) när A-registret skall laddas med ett nytt datavärde från bussen. Vid alla andra tillfällen skall den vara passiv (=0).

Samtliga instruktioner som laddar register A måste därför generera en etta på OR-grindens utgång under det tillstånd då laddningen skall ske.



Motsvarande Booleska uttryck:
$LD_A = Q_5 \cdot I_{02} + Q_5 \cdot I_{04} + Q_6 \cdot I_{07} + \ldots + Q_9 \cdot I_{AD}$

Figur 7 Logiknät och motsvarande Booleska uttryck för styrsignalen LD_A.

Av figur 7 kan man se att den översta AND-grinden (och därmed OR-grinden som bildar styrsignalen LD_A) ger utsignalen 1 om $Q_5 \cdot I_{02} = 1$.

Register A laddas alltså när styrenheten befinner sig i tillstånd Q_5 och OP-koden 02H samtidigt finns i instruktionsregistret. Eftersom Q_5 är det första tillståndet i EXECUTE laddas register A i det första tillståndet i EXECUTE för instruktionen med OP-koden 02H.

Nästa AND-grind ger utsignalen 1 om $Q_5 \cdot I_{04} = 1$.
Register A laddas när styrenheten befinner sig i tillstånd Q_5 och OP-koden 04H samtidigt finns i instruktionsregistret. Här laddas register A i det första tillståndet i EXECUTE för instruktionen med OP-kod 04H.

Följande AND-grind ger utsignalen 1 om $Q_6 \cdot I_{07} = 1$.
Register A laddas när styrenheten befinner sig i tillstånd Q_6 och OP-koden 07H samtidigt finns i instruktionsregistret. Här laddas register A i det andra tillståndet i EXECUTE för instruktionen med OP-kod 07H.

Den nedersta AND-grinden (och OR-grinden) ger utsignalen 1 om $Q_9 \cdot I_{AD} = 1$.
Register A laddas när styrenheten befinner sig i tillstånd Q_9, d v s det femte tillståndet i EXECUTE för instruktionen med OP-kod ADH.

Eftersom A-registret endast laddas i EXECUTE-fasen (av de instruktioner som påverkar A), består samtliga termer i summan som bildar LD_A av en produkt $Q_i \cdot I_j$, där Q_i (i = 5, …, FH), är en av utsignalerna från tillståndsavkodaren i figur 6 och I_j, (j = 0, …, FFH), är en av utsignalerna från instruktionsavkodaren i samma figur.

Implementering av styrenhet för fyra instruktioner

För att illustrera principen för styrenheten i föregående avsnitt skall vi nu konstruera en styrenhet som klarar de fyra instruktioner som ingår i programmet nedan

```
        LDAA    #$90
LOOP    INCA
        STAA    $05
        JMP     LOOP
```

Styrenheten skall alltså klara av RESET-sekvensen (för start av programmet), FETCH-sekvensen och EXECUTE-sekvenserna för de fyra instruktionerna i programmet.

En lämplig utgångspunkt vid konstruktionen är att beskriva samtliga sex sekvenser som behövs med RTN och styrsignaler.

RESET- och FETCH-sekvensen har beskrivits tidigare och återges i tabellform nedan.

RESET:

Tillstånd	Summaterm	RTN-beskrivning	Styrsignaler (=1)
Q_0	Q_0	FFH→R	f_3, f_2, f_1, f_0, LD_R
Q_1	Q_1	R→MA	OE_R, LD_{MA}
Q_2	Q_2	M→PC	MR, LD_{PC}

FETCH:

Tillstånd	Summaterm	RTN-beskrivning	Styrsignaler (=1)
Q_3	Q_3	PC→MA, PC+1→PC	OE_{PC}, LD_{MA}, IncPC
Q_4	Q_4	M→I	MR, LD_I

EXECUTE-sekvensen för LDAA #Data visas nedan.

EXECUTE för LDAA #Data: (OP-kod = 0FH)

Tillstånd	Summaterm	RTN-beskrivning	Styrsignaler (=1)
Q_5	$Q_5 \cdot I_{0F}$	PC→MA, PC+1→PC	OE_{PC}, LD_{MA}, IncPC
Q_6	$Q_6 \cdot I_{0F}$	M→A, (Next Fetch)	MR, LD_A, NF

I kolumnen *Tillstånd* visas styrenhetens tillstånd. I tabellerna ingår den nya kolumnen *Summaterm*, som visar summaterm $Q_i \cdot I_j$ enligt principen i figur 7.

Ext-14

Övningsuppgift 1a: Fyll i RTN-beskrivning och styrsignaler för EXECUTE-sekvensen för instruktionen INCA i tabellen nedan. Observera att det första tillståndet i EXECUTE alltid är Q_5 och att styrsignalen NF alltid skall vara aktiv i det sista tillståndet i EXECUTE. Tänk också på att INCA skall påverka flaggorna!

EXECUTE för INCA: (OP-kod = 41H)

Tillstånd	Summaterm	RTN-beskrivning	Styrsignaler (=1)
Q_5	$Q_5 \cdot I_{41}$		
Q_6	$Q_6 \cdot I_{41}$		

¤¤¤

Övningsuppgift 1b: Fyll i OP-kod, Tillstånd, Summaterm, RTN-beskrivning och Styrsignaler i tabellerna nedan. Observera att det första tillståndet i EXECUTE alltid är Q_5.

Styrenheten skall alltid återvända till första tillståndet i FETCH efter det sista tillståndet i EXECUTE. Därför måste man sätta NF=1 i det sista tillståndet i EXECUTE.

Antalet klockcykler (states) för EXCUTE-sekvenserna får man fram genom att slå upp antalet states för respektive instruktion i instruktionslistan för FLEX-processorn och sedan dra bort antalet states i FETCH-sekvensen. På detta sätt får man 5 - 2 = 3 för STAA Adr och 4 - 2 = 2 för JMP Adr

EXECUTE för STAA Adr: (OP-kod = H)

Tillstånd	Summaterm	RTN-beskrivning	Styrsignaler (=1)

EXECUTE för JMP Adr: (OP-kod = H)

Tillstånd	Summaterm	RTN-beskrivning	Styrsignaler (=1)

¤¤¤

Övningsuppgift 1c:
Översätt instruktionssekvensen nedan till maskinkod med hjälp av instruktionslistan för FLEX-processorn. Instruktionssekvensens startadress i minnet skall vara 20H.

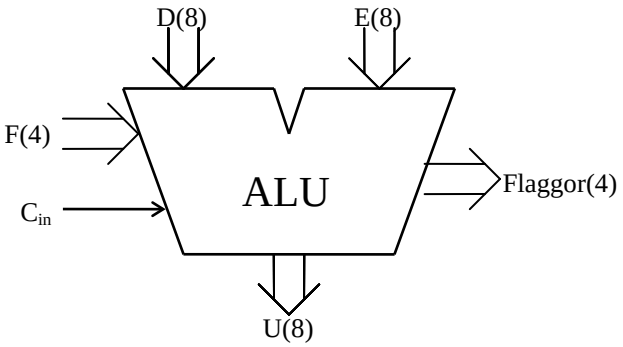
Adress (Hex)	
1F	-
20	
21	
22	
23	
24	
25	
26	
27	

```
        LDAA    #$90
LOOP    INCA
        STAA    $05
        JMP     LOOP
```

¤¤¤

Bilaga 1

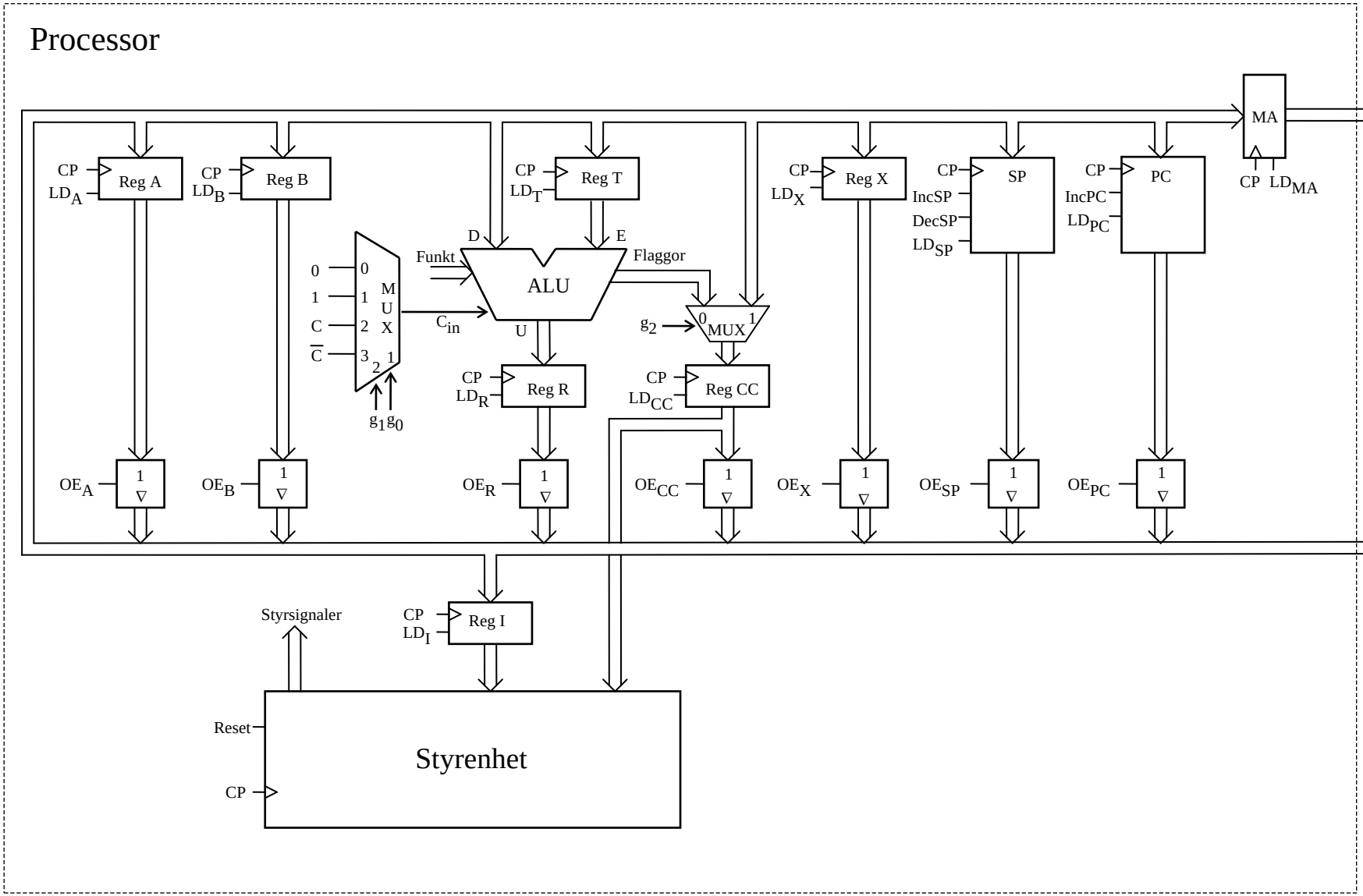
ALU:ns funktion



ALU:ns logiska och aritmetiska operationer på indata D och E definieras av ingångarna $F = (f_3, f_2, f_1, f_0)$ och C_{in} enligt tabellen nedan.

f_3 f_2 f_1 f_0	$U = f(D,E,C_{in})$
0 0 0 0	0
0 0 0 1	D
0 0 1 0	E
0 0 1 1	D'
0 1 0 0	E'
0 1 0 1	D OR E
0 1 1 0	D AND E
0 1 1 1	D XOR E
1 0 0 0	$D + C_{in}$
1 0 0 1	$D - 1 + C_{in}$
1 0 1 0	$D + E + C_{in}$
1 0 1 1	$D + D + C_{in}$
1 1 0 0	$D - E - 1 + C_{in}$
1 1 0 1	0
1 1 1 0	0
1 1 1 1	FFH

I tabellen ovan avser "+" och "-" aritmetiska operationer. Med t ex D' menas att samtliga bitar i D inverteras.



Figur 1. Datorn FLEX.