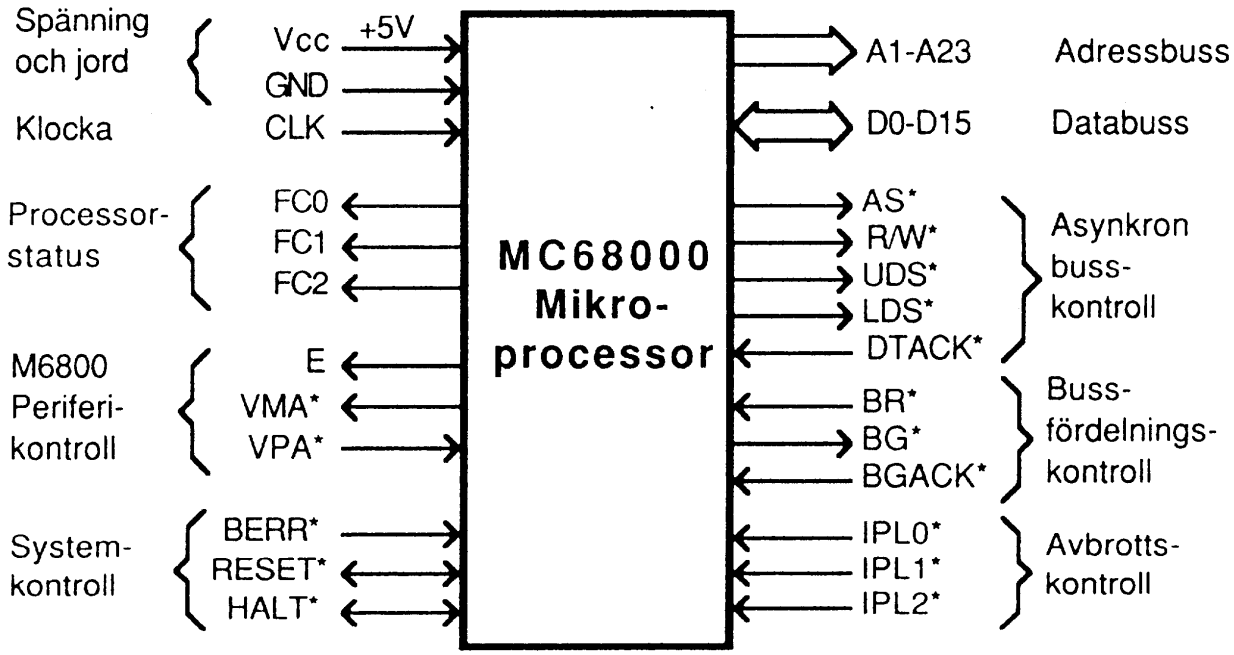
F2: Motorola 68000

- 68000
  - Arkitektur
    - I/O signaler
    - Processor arkitektur
    - Programmeringsmodell
  - Assembler vs. Maskinkod
  - Exekvering av instruktioner i 68000
  - Instruktionsformat
  - MOVE instruktionen
    - Adresseringsmoder

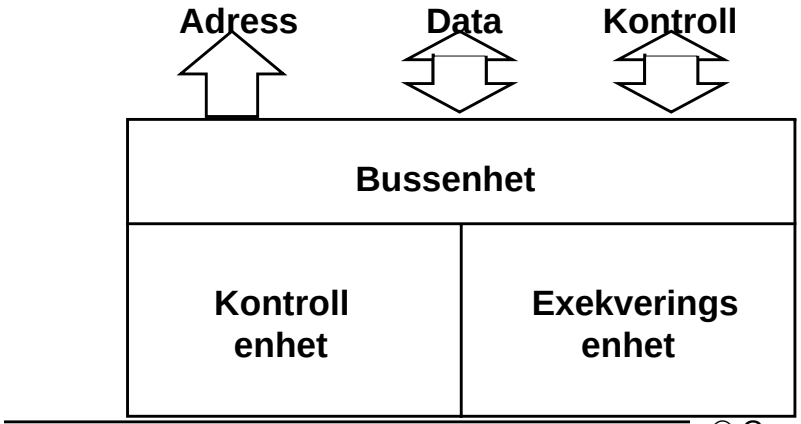
MC68000

- **Inre arkitektur som är 32-bits (logiskt)**
- **Yttre**
  - **16 - bit databuss**
  - **24 - bit adressbuss, 16 MBytes kan adresseras**



Arkitektur MC68000

- **Exekveringsenhet**
  - **Utför logiska/aritmetiska operationer**
  - **Minnesenhet, registerarea**
- **Kontrollenhet**
  - **Översätter instruktionerna till handling**
- **Bussenhet**
  - **Kontrollerar läs/skriv data operationer**
  - **Hämtar instruktioner från minnet**

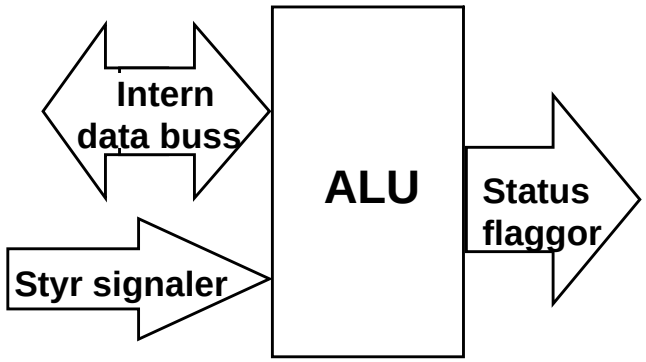


68000, Bussenhet

- **24 - bitars bred adressbuss**
  - **16 Mbytes,(1 kByte = 2^{10} bytes, 1 Mbytes = 2^{20} bytes)**
- **16 - bitars bred databuss**
- **2 kontrollbussar**
  - **Asynkron kontroll**
    - **Synkronisering m.h.a. handskakning**
  - **Synkron kontroll (MC6800 buss)**
    - **Emulerar MC6800 buss kontroll**
    - **Saknar handskakning**

68000, Exekveringsenhet

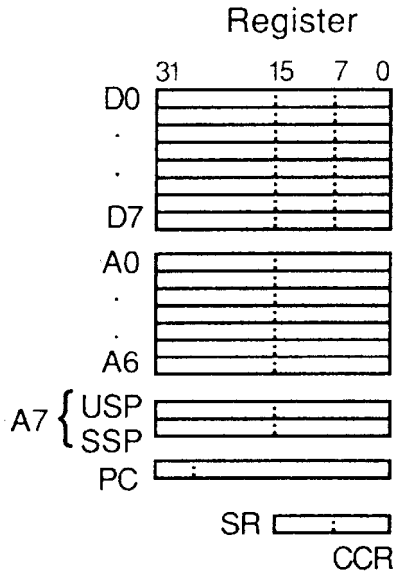
- **Består av Register och ALU**
  - **16 allmänna register**
  - **ALU, Arithmetic Logic Unit**
    - **Denna enhet utför beräkningar och logiska operationer på innehållet i processorns register.**
  - **En ALU är uppbyggd med digitala grindar som utför aritmetiska operationer**



Innehåller även osynliga register

Register i MC68000

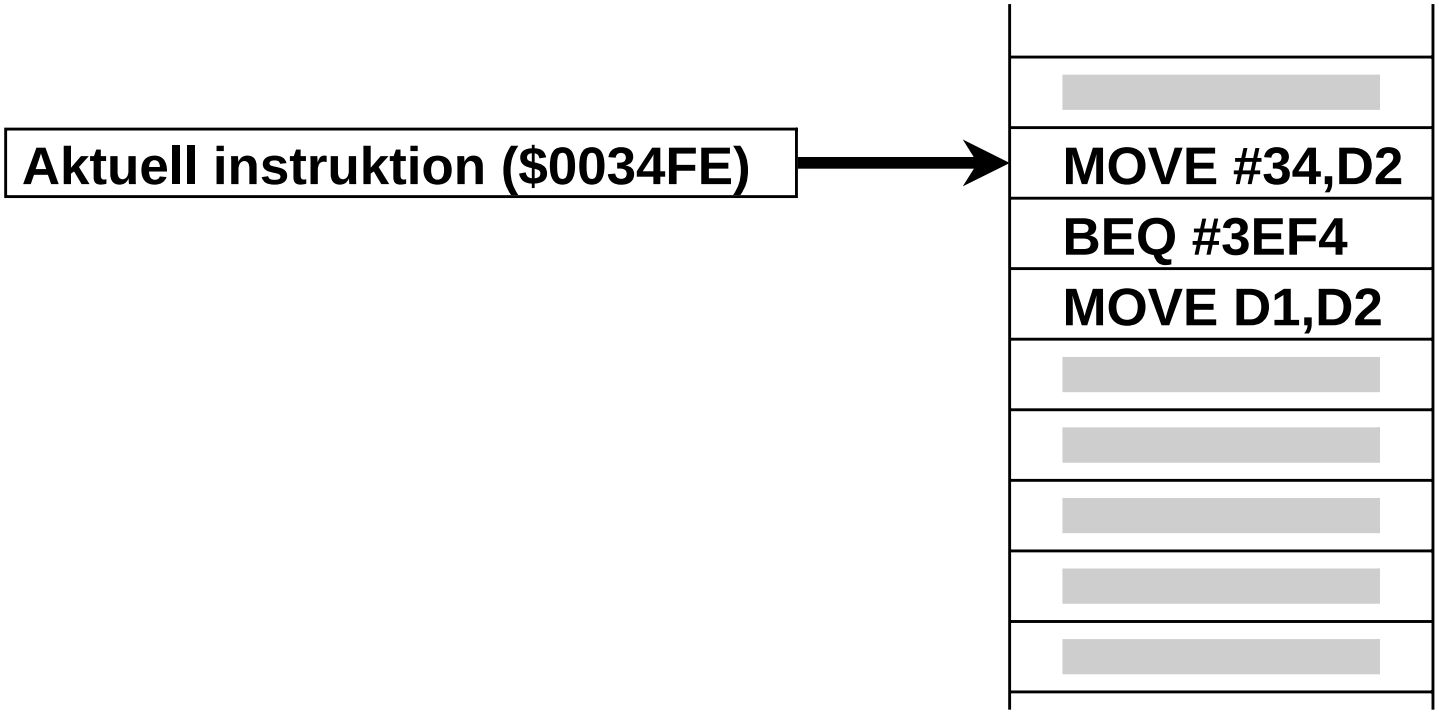
- 8 - dataregister, D0 - D7
- 7 - adressregister, A0 - A6
- 1 - Stackpekare, A7
- 1 - Condition Code Register, CCR (del av SR)
- 1 - Programräknare, PC



Mittuniversitetet
MID SWEDEN UNIVERSITY

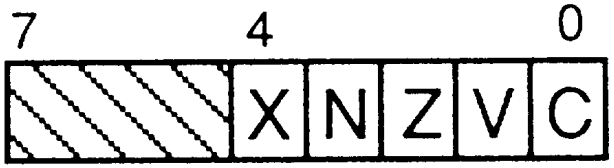
PC

- **Programräknare**
  - **Pekar vart i minnet den exekverade instruktionen finns**



CCR

- **Bitarna i registret påverkas av dom olika instruktionerna.**
- **Titta i instruktionsmanualen efter hur varje instruktion påverkar varje ‘flagga’**



X = Extended (= ”äkta carry”, för räkning i multipel precision)
N = Negative
Z = Zero
V = Overflow
C = Carry

Exempel på påverkan av CCR

Table 2.1 Effects of various instructions on the CCR

	ADD.B	ADD.B	SUB.B	SUB.B	ASL.B	ASL.B	ASR.B	ASR.B	CLR.B
Source	00101011	01101011	00011010	01011010	00101011	01101011	10101011	01101011	00101011
Destination	00011010	01011010	00101011	00011011					
Destination	01000101	11000101	00010001	11000001	01010110	11010110	11010101	00110101	00000000
CCR	XNZVC	XNZVC	XNZVC	XNZVC	XNZVC	XNZVC	XNZVC	XNZVC	XNZVC
	00000	01010	00000	11001	00000	01010	11001	10001	-0100

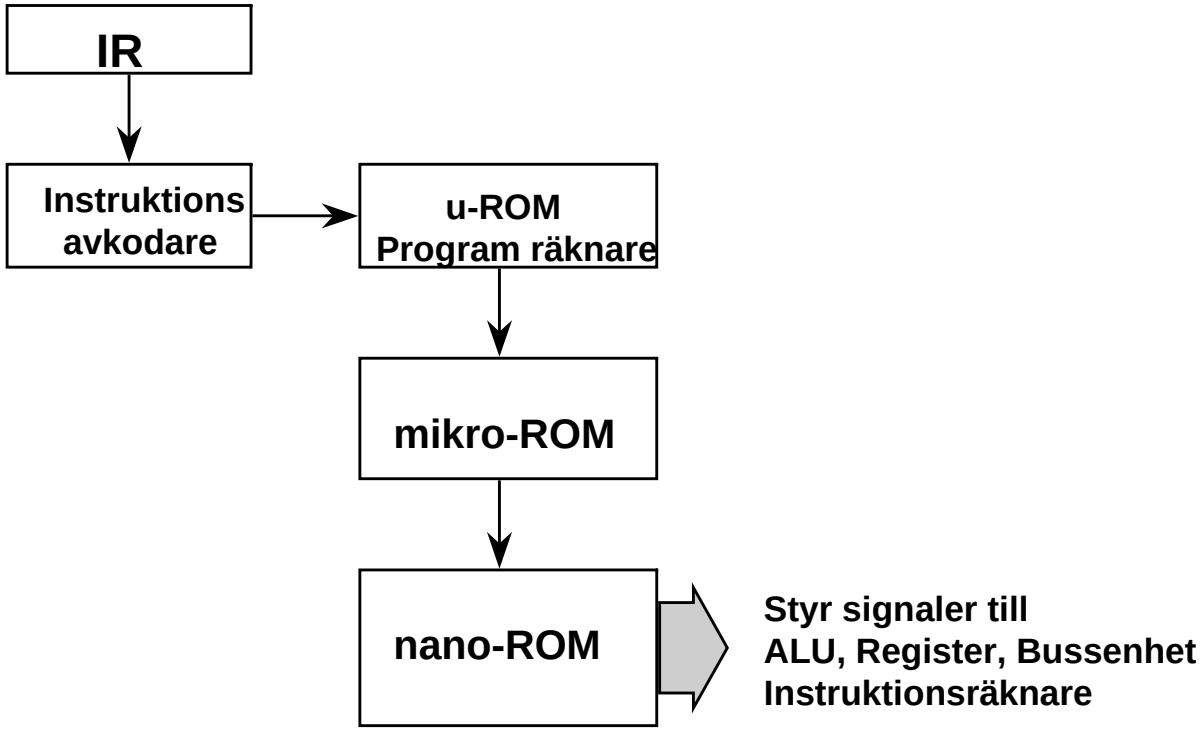
$43+26=69_D$ **$107+90=197_D$** **$43-26=17_D$** **$27-90=-63_D$**

Overflow därför att talet 197 ej kan representeras med en byte i 2-komplement

Exekvering av 68000 instruktioner

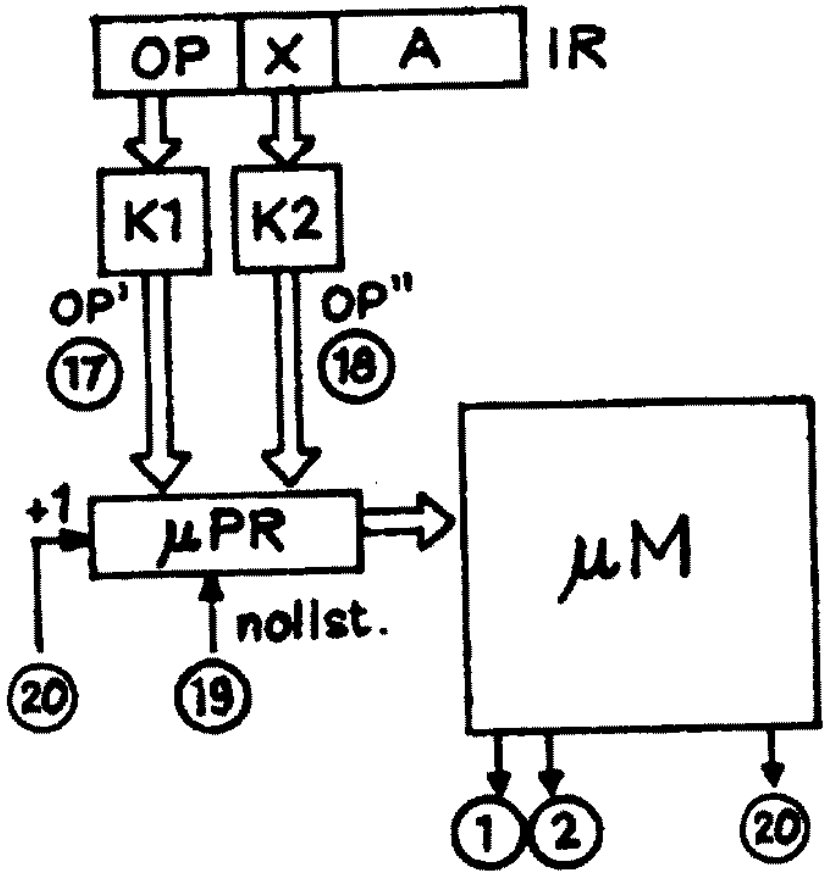
- **Bussenheten laddar IR (Instruction Register), med en ny operations kod.**
  - **Varje instruktion (operationskod) pekar ut ett litet program (mikro program) i ett μ-ROM i processorn.**
- **Instruktions avkodaren översätter operationskoden till ett värde på u-kods programräknaren.**
  - **u-PC pekar på ett mikroprogram som utför instruktionen**
  - **För varje klockpuls räknar μPC upp och μ-kods ROM:et ger nytt värde (adress) till ett nano-ROM.**
  - **För varje värde klockpuls ger nano-ROM:et ut en kombination av styrsignal till alla enheter i CPU:n.**

Exekvering av mikroprogram/instruktioner

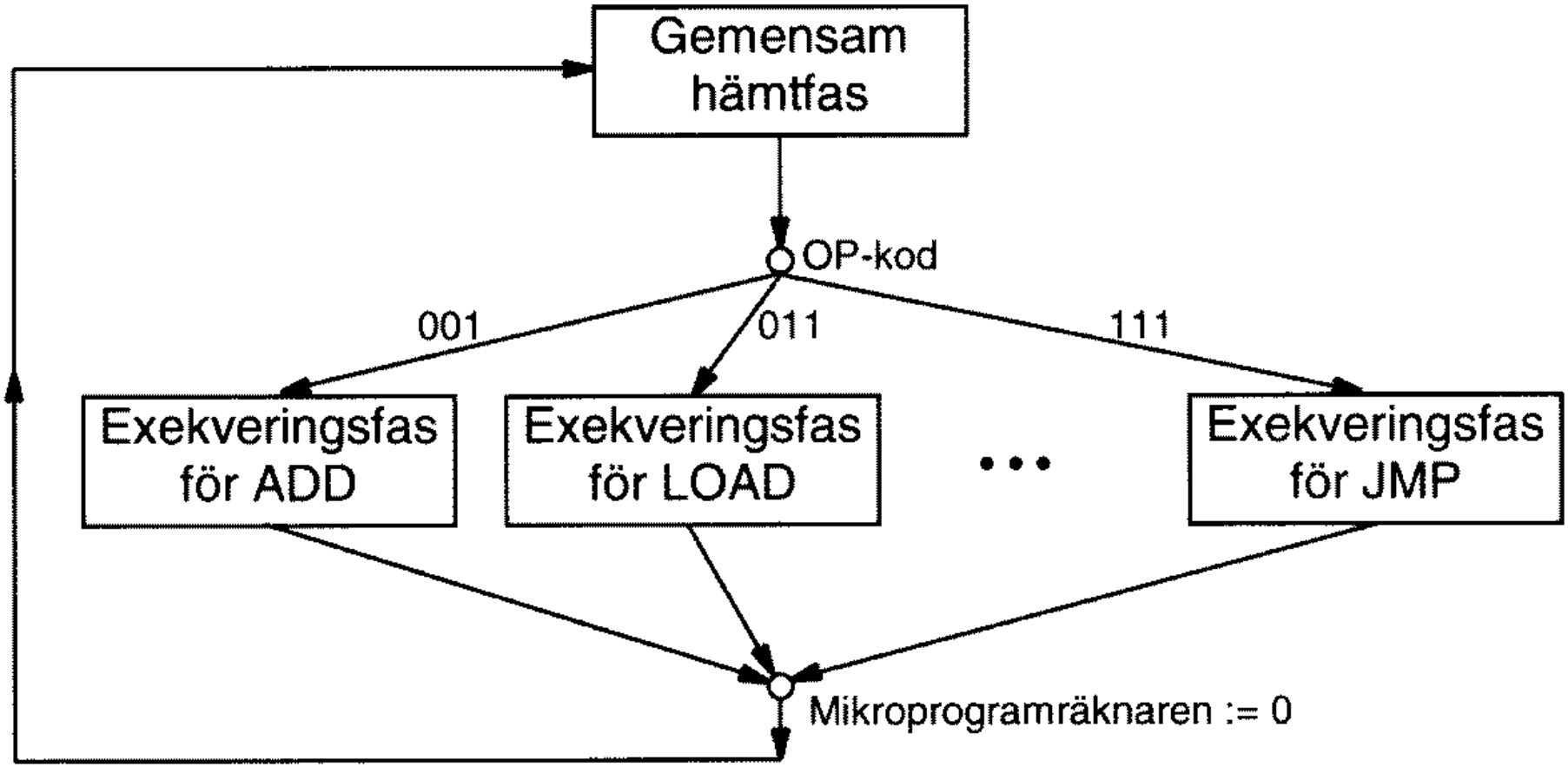


Instr.	OP-kod	Startadress till mikroprogramminnet						
ADD	001	1	1	1	0	0	0	1
LOAD	011	1	0	0	1	0	1	1

Tabell 3. Tabell för översättning av operationskod till startadress



Exekvering av mikroprogram/instruktioner

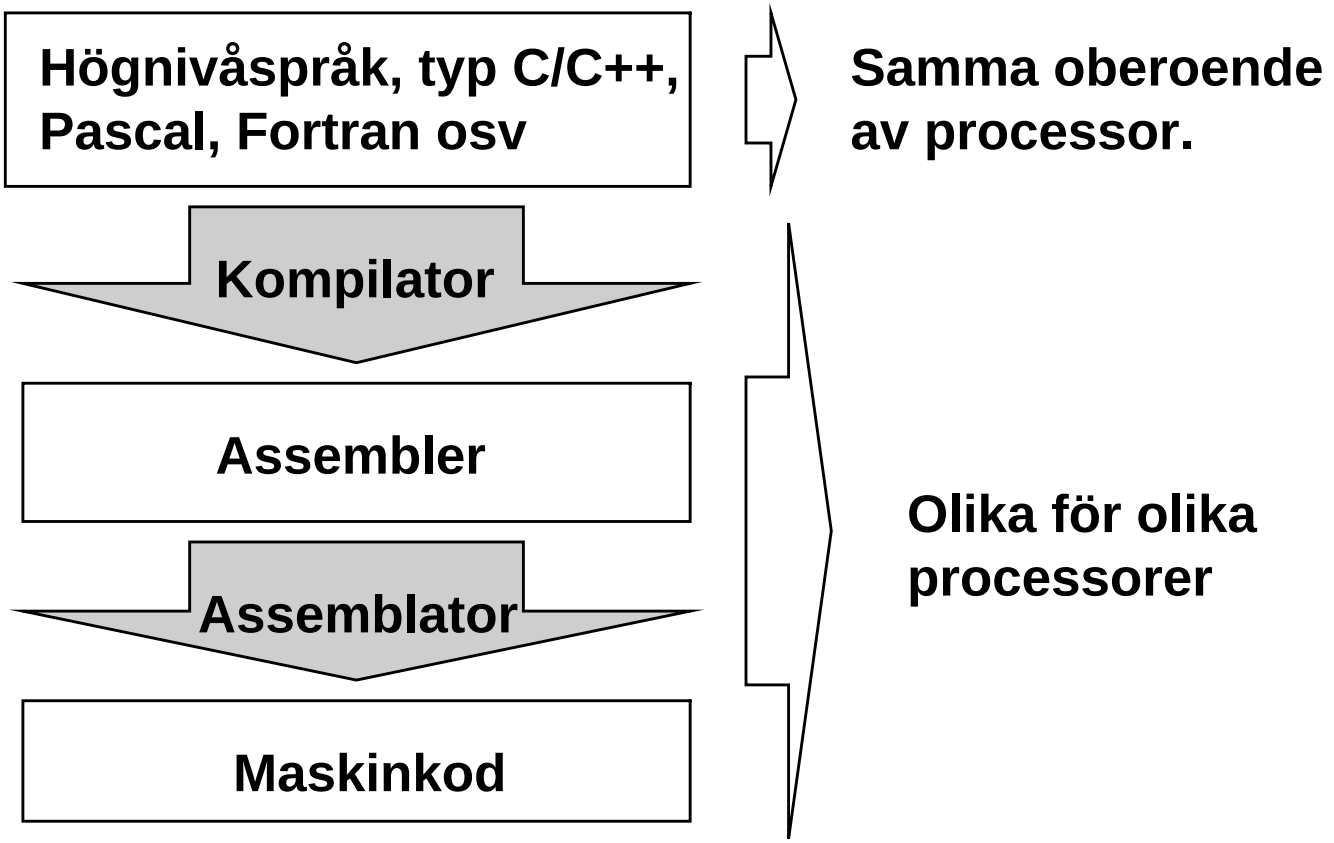


Vad är maskinkod?

- **Den sekvens av binära koder, som utgör ett program.**
  - **Mycket svårt att förstå**
  - **Ex.**
    - **111000000000110**
- **Assembler är en översättning av maskinkoden så att människan ska förstå.**
  - **Ex.**
    - **MOVEQ #6, D0 (Samma sak som ovan)**

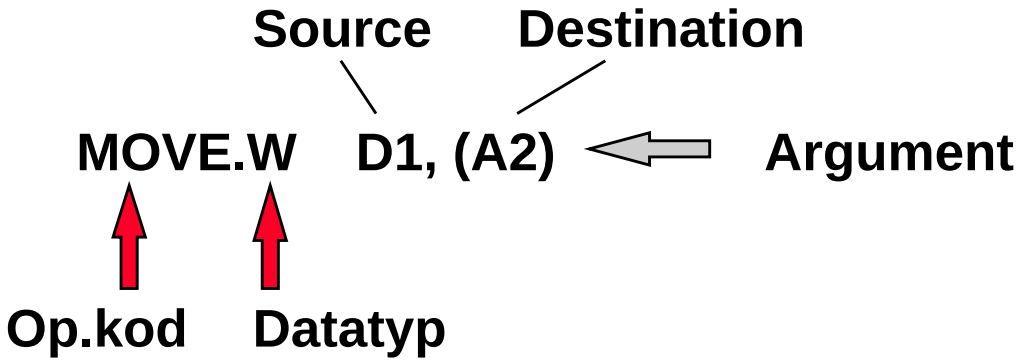
Programmering i assembler

- **Assembler är en textversion av maskinkoden**



68000, Assembler

- **En assemblerinstruktion består av**
  - **Operationskod**
    - Matematiska/ Logiska och dataflyttningsoperationer.
  - **Datatyp**
    - B = Byte, 8 bit.
    - W = Word, 16 bit.
    - L = Long word, 32 bit.
  - **Argument**
    - Dataregister
    - Minnesposition
    - Adressregister
    - Osv..

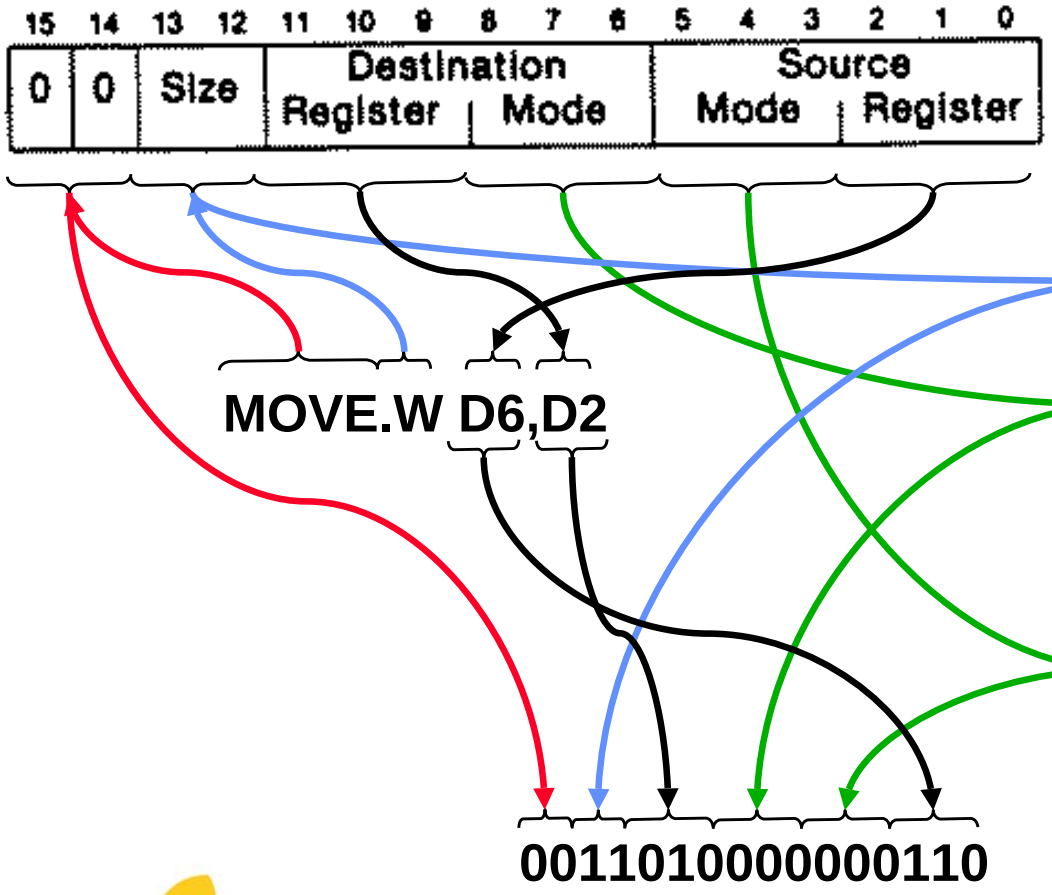


Mittuniversitetet
MID SWEDEN UNIVERSITY

MOVE - instruktionen.

- **MOVE**
  - **Kopierar data från ett ställe till ett annat**
  - **<Source> -> <Destination>**
  - **Ex.**
    - `MOVE #43, D0`
- **MOVEA**
  - **Samma som MOVE men destinationen är ett adressregister.**
  - **<Source> -> <An>**
  - **Ex.**
    - `MOVEA D3, A4`

Översättning från assembler till maskin kod



Size field — Specifies the size of the operand to be moved:
- 01 — byte operation.
- 11 — word operation.
- 10 — long operation.

Addressing Mode	Mode	Register
Dn	000	register number
An	—	—
(An)	010	register number
(An) +	011	register number
– (An)	100	register number
d(An)	101	register number

Addressing Mode	Mode	Register
Dn	000	register number
An*	001	register number
(An)	010	register number
(An) +	011	register number
– (An)	100	register number
d(An)	101	register number

Adresseringsmetoder

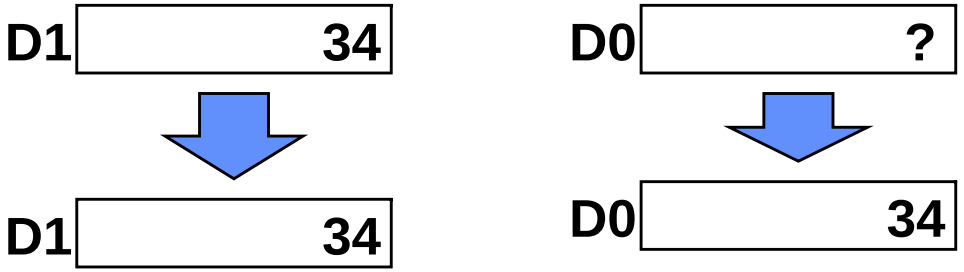
- **Behandlar hur data läses och skrivs i instruktioner**
- **Olika metoder**
  - **Direkt register adressering (Direct register addressing)**
  - **Direkt data adressering (Immediate addressing)**
  - **Absolut adressering (Absolute addressing)**
  - ***Indirekt adressering***

Direkt register adressering

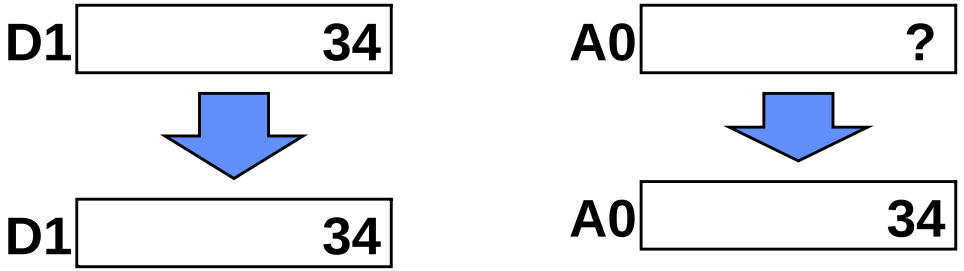
- **Enklast**
  - **En operand är ett dataregister eller adressregister**

Dataregister:
MOVE D1,D0

källoperand
destinationsoperand



Adressregister:
MOVEA D1,A0

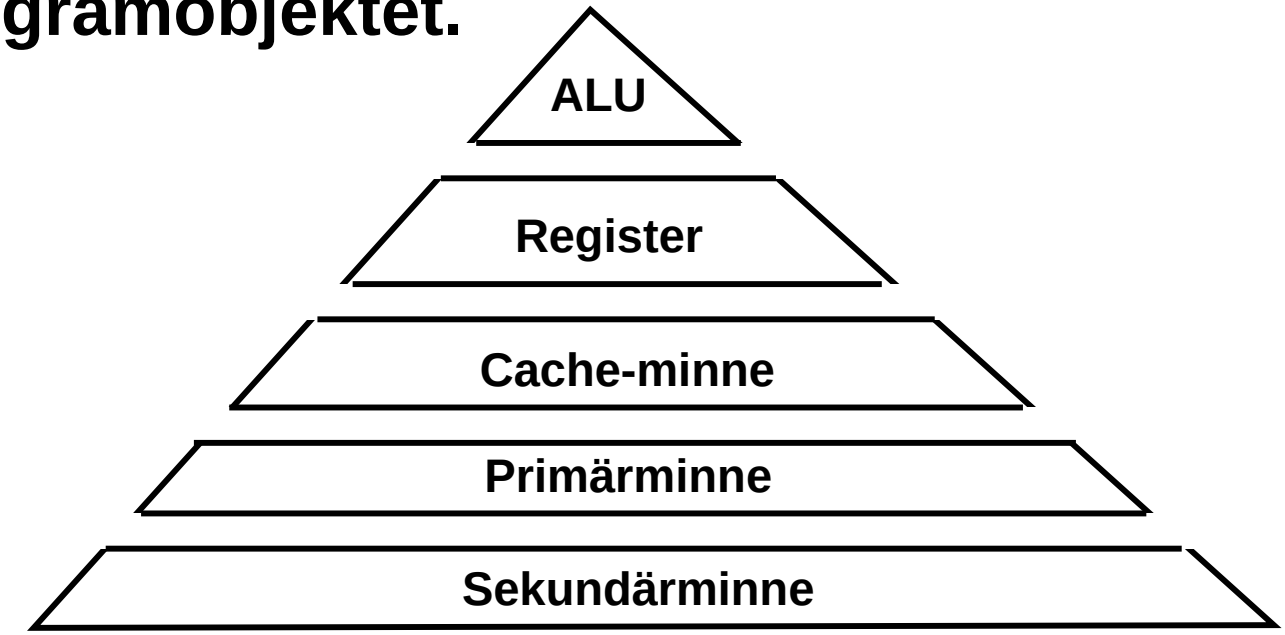


MOVE.B D0,D3	Copy the source operand in register D0 to register D3
SUB.L A0,D3	Subtract the source operand in register A0 from register D3
CMP.W D2,D0	Compare the source operand in register D2 with register D0
ADD D3,D4	Add the source operand in register D3 to register D4

Mittuniversitetet
MID SWEDEN UNIVERSITY

Direkt adressering

- **Direkt adressering, resulterar i korta instruktioner**
  - **Eftersom man behöver bara 3 bitar för att definiera ett register.**
- **Det är en snabb instruktion**
  - **Eftersom inget externtminne behöver accessas.**
- **Som programmerare bör man använda direkt adressering för att lagra variabler som används ofta i programobjektet.**



Direkt data

- **Operanden är en del av instruktionen**
- **’#’ används för att indikera ’direkt data’ adressering**
- **Till exempel**
  - **MOVE.B #24,D0 använder direkt data operanden 24.**

Före

D0 ?

Efter

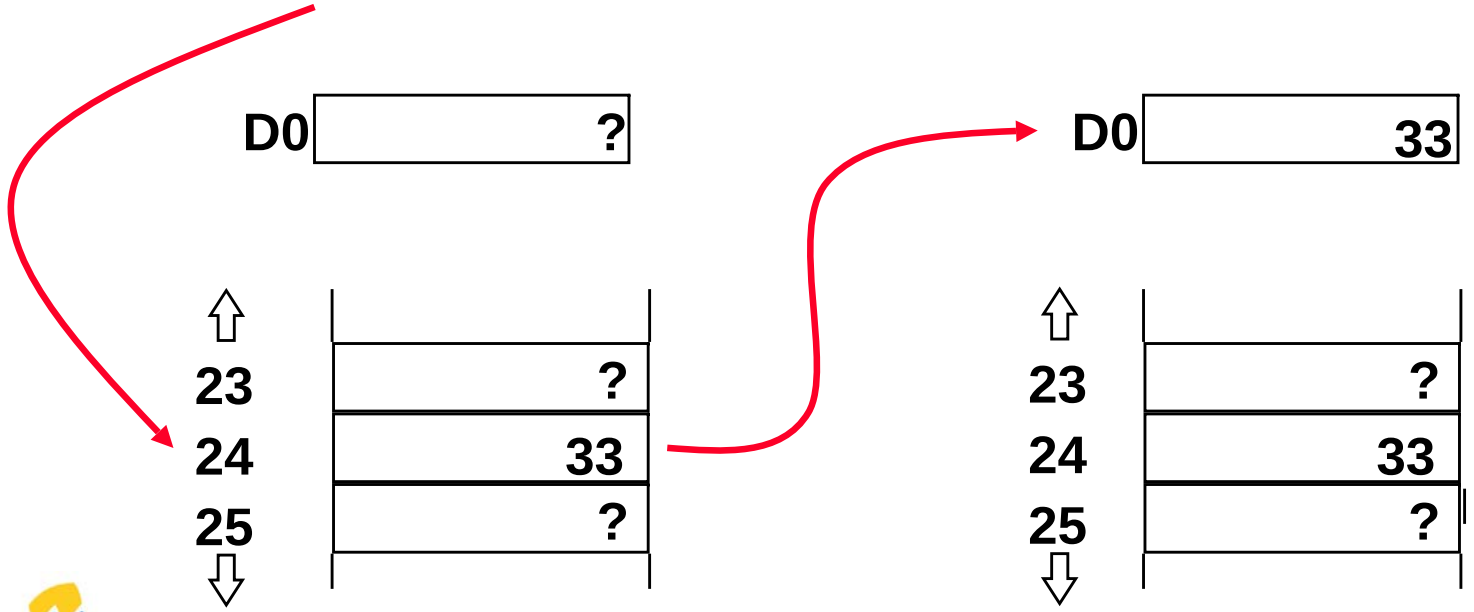
D0 24

Absolut adressering

- **Kopiera innehållet från/till den specificerade adressen.**
  - **MOVE.W D0, adr.W**
  - **MOVE.W D0, adr.L**

  - **Ex. MOVE.B 24,D0**
    - **Före:**

Efter:



Direkt data / Absolut adressering

- **För direkt data och absolut adressering finns data eller adressen till data inbakad i instruktionen.**
- **Absolut adressering kräver 2 (3) minnes accesser vid exekvering**
  - **1. Hämta instruktionen**
  - **2. Hämta operanden (adressen till operanden)**
- **Exempel**
  - **ADD #3,D0**
    - **Adderar 3 till D0**
  - **ADD 3,D0**
    - **Adderar innehållet i minnes position 3 med D0**

Sammanfattning av grundläggande adressering

- **Studera följande högnivå exempel:**

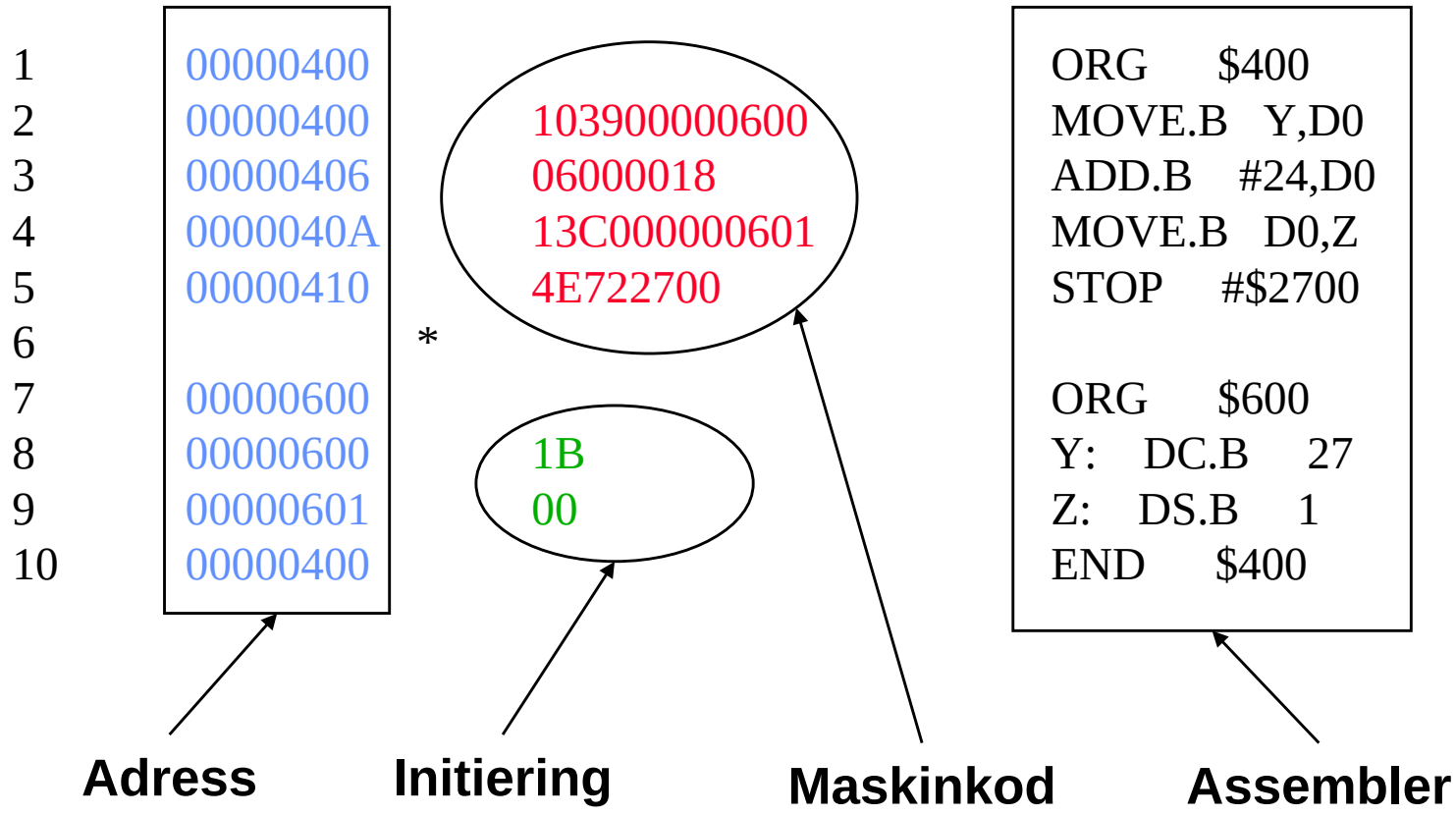
  Z = Y + 4

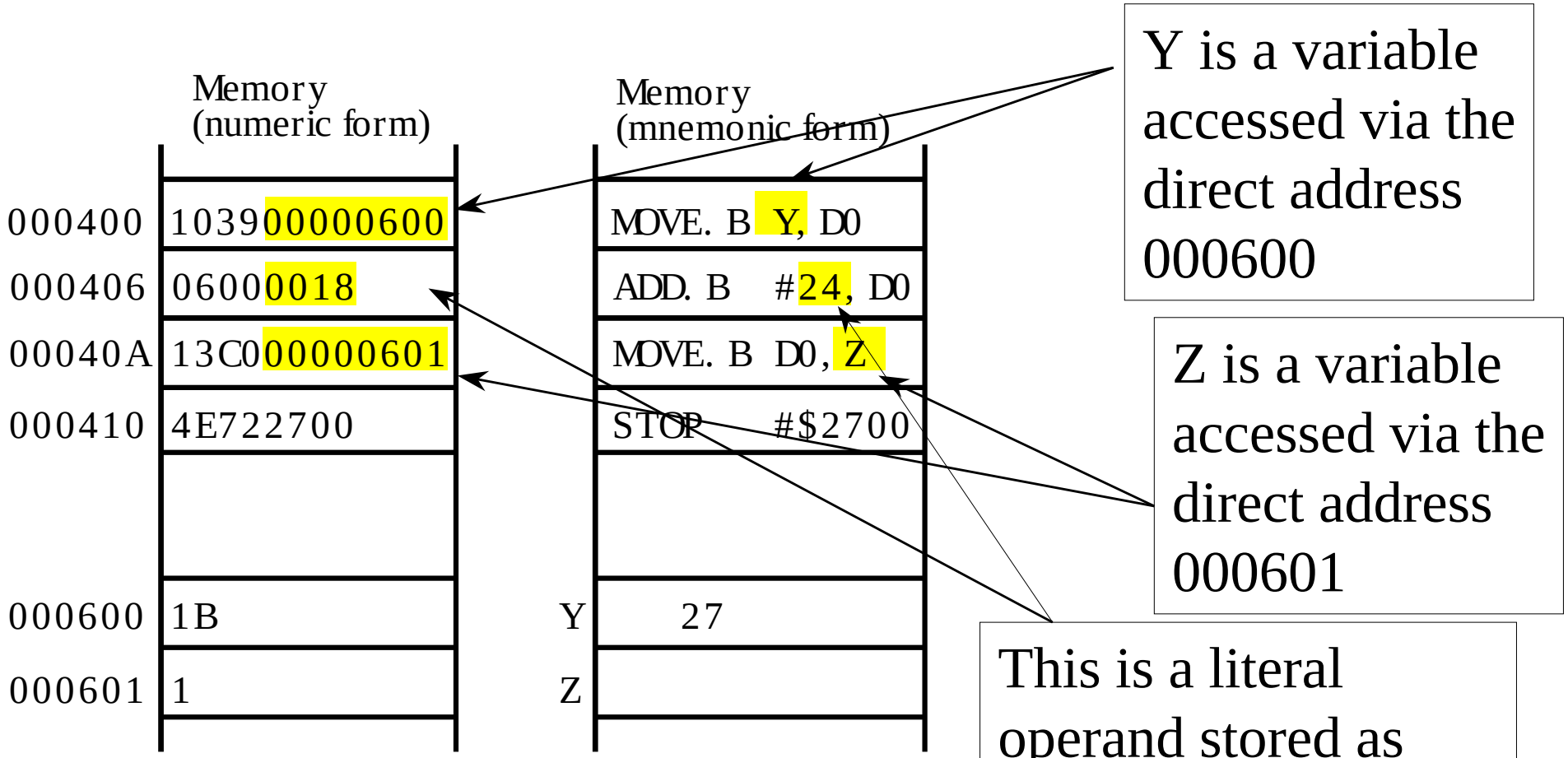
- **Följande kod implementerar funktionen**

```
        ORG    $400      Start of code
        MOVE.B Y,D0
        ADD    #4,D0
        MOVE.B D0,Z

        ORG    $600      Start of data area
Y       DC.B   27        Store the constant 27 in memory
Z       DS.B   1         Reserve a byte for Z
```

Den assemblerade koden





Sammanfattning

- **Register direkt adressering används för variabler som kan lagras i register**
- **Direkt data adressering används för konstanter**
- **Absolut adressering används för variabler som måste lagras i minne eller I/O portar**
- **Den enda skillnaden mellan register direkt adressering och absolut adressering är att den första använder ett register för att lagra operanden och den senare använder en minnesposition.**