HUMBOLDT-UNIVERSITÄT ZU BERLIN

Technische Informatik 2
Adressierungsarten

Prof. Dr. Miroslaw Malek

Sommersemester 2007

www.informatik.hu-berlin.de/rok/ca

Thema heute

- X-Adressrechner
  - 0-Adressrechner
  - 1-Adressrechner
  - 2-Adressrechner
  - 3-Adressrechner
- Adressierungsarten
  - Von der Absoluten zur Blockadressierung
- Beispiele
  - PDP-11
  - PowerPC
  - Pentium

0-Adressrechner

- Den Befehlen werden keine Adressen von Operanden übergeben
  → es gibt nur *eine* feste Adresse
  (letzte Elemente auf Stack)
- Bekannt als Stackmachine (z.B. HP-3000)

Befehl	Kommentar (Ergebnis: X = A * B + C * C)
PUSH A	Transferiert A auf den Stapel
PUSH B	Transferiert B auf den Stapel
MULTIPLY	Holt A, B vom Stapel und ersetzt sie durch A * B
PUSH C	Transferiert C auf den Stapel
DUP	Verdoppelt letztes Element auf dem Stapel
MULTIPLY	Holt C, C vom Stapel und ersetzt sie durch C * C
ADD	Holt C*C, A*B vom Stapel, ersetzt sie durch Summe
POP X	Transferiert Ergebnis vom Stapel nach X

1-Adressrechner

- Es gibt nur ein Daten-Register:
  Der Akkumulator
- Befehle bekommen nur maximal eine Adresse übergeben, der 2. Operand befindet sich im Akkumulator
- Das Ergebnis wird auch dorthin gespeichert

Befehl	**Kommentar (Ergebnis: X = A * B + C * C)**
LOAD A	**Transferiert A in den Akkumulator AC**
MULTIPLY B	**AC $\longleftarrow$ AC * B**
STORE T	**Transferiert AC zur Speicherstelle T**
LOAD C	**Transferiert C in den Akkumulator AC**
MULTIPLY C	**AC $\longleftarrow$ AC * C**
ADD T	**AC $\longleftarrow$ AC + T**
STORE X	**Transferiert Ergebnis zur Speicherstelle X**

2-Adressrechner

- Befehl enthält 2 Adressen
- 2. Adresse ist meist auch Zieladresse (Ergebnis zurückschreiben)
- Ganz verbreitet

Befehl	Kommentar
MOVE A, R0	R0 ← A
MULTIPLY B, R0	R0 ← R0 * B
MOVE C, R1	R1 ← C
MULTIPLY C, R1	R1 ← R1 * C
ADD R0, R1	R1 ← R0 + R1
MOVE R1, X	X ← R1

3-Adressrechner

- Befehl enthält 3 Adressen meist der Form: Quelle1, Quelle2, Ziel
- ermöglicht sehr kurze Programme

Befehl	Kommentar
MULTIPLY A, B,T	$T \leftarrow A * B$
MULTIPLY C, C, X	$X \leftarrow C * C$
ADD X, T, X	$X \leftarrow X + T$

Tradeoffs

	Verschiedene Adressmaschinen			
Adressen	0	1	2	3
Aufteilung von 32 Bit	32 Bit OpCode/ Daten	8 Bit 24 Bit O Adresse	8 12 12 O A1 A2	8 8 8 8 O A1 A2 A3
Verarbeitungs-geschwindig-keit/ Befehl	Sehr hoch	Hoch	Normal	Gering
Befehle im Beispiel	8	7	6	3
Befehlsanzahl	Sehr viele	Viele	Wenige	Sehr wenige

Allgemeine Adressierungsmethoden

- Absolute (Direkte) Adressierung

  Die Adresse des Operanden ist explizit als Teil des Befehles angegeben.

- Implizite Adressierung (implied)

  Die Adresse ergibt sich aus dem Befehl (z. B. wird in einem Einadressrechner der Akkumulator als Adresse des zweiten Operanden angesehen).

- Unmittelbare Adressierung (immediate)

  Der Operand wird explizit im Befehl angegeben. Es ist kein Speicherzugriff erforderlich. Der Operand kann auch direkt nach dem Befehl stehen.

Allgemeine Adressierungsmethoden (2)

- Indirekte Adressierung (indirect)

  Die effektive Adresse des Operanden steht in dem Register oder Hauptspeicherplatz, dessen Adresse im Befehl angegeben ist. Dies kann in mehreren Stufen erfolgen.

- Indizierte Adressierung (indexed)

  Die effektive Adresse (EA) des Operanden wird aus der Addition des Wertes eines Indexregister (X) und der direkten Adresse (DA) berechnet.

  $$EA = X + DA$$

- Basis Adressierung (base)

  Die effektive Adresse des Operanden wird berechnet durch die Addition des Wertes eines Basisregisters und der direkten Adresse.

  $$EA = B + DA$$

Allgemeine Adressierungsmethoden (3)

- Relative Adressierung (self-relative, relative)

  Die effektive Adresse ist die Summe der direkten Adresse und des Inhalts des Befehlszählers (PC).

  $$EA = DA + PC$$

- Segmentierte Adressierung (augmented)

  Die effektive Adresse ergibt sich aus dem Zusammenfügen des Inhalts des Segment-Adressregisters (SAR) und der direkten Adresse.

  $$EA = SAR \; || \; DA$$

  (z.B. SAR spezifiziert dabei eine Speicherseite und DA ist eine Adresse innerhalb dieser speziellen Seite.)

Allgemeine Adressierungsmethoden (4)

- Block Adressierung

  Die Adresse des ersten Wortes im Block ist gegeben.

  Die Anzahl der Wörter wird gewöhnlich durch:

  - den Befehl bestimmt oder
  - es kann auch die letzte Adresse angegeben werden oder
  - ein besonderes end-of-block Zeichen oder
  - Blöcke können eine feste Länge haben

  Sehr nützlich bei der Verwaltung des Sekundärspeichers.

Adressierung am Beispiel der PDP-11

PDP-11: Adressierungsarten

$B_5B_4B_3$	Dezimal	Name	Syntax	Bedeutung
000	0	Register	Rn	EA = R_n(d.h., Operand = [R_n])
010	2	Autoincrement	(Rn)+	EA = [R_n] Erhöhe R_n
100	4	Autodecrement	- (Rn)	Verringere R_n EA = [R_n]
110	6	Index	X(Rn)	Hole X; Erhöhe PC EA = X + [R_n]
001	1	Register Indirect	@ Rn	EA = [R_n]
011	3	Autoincrement Indirect	@(Rn)+	EA = [[R_n]] Erhöhe R_n

EA = Effektive Adresse
[Z] = Inhalt von Z
[[Z]] = Inhalt der Speicherzelle, auf die Z zeigt.

OP-Code	Modus	R_n

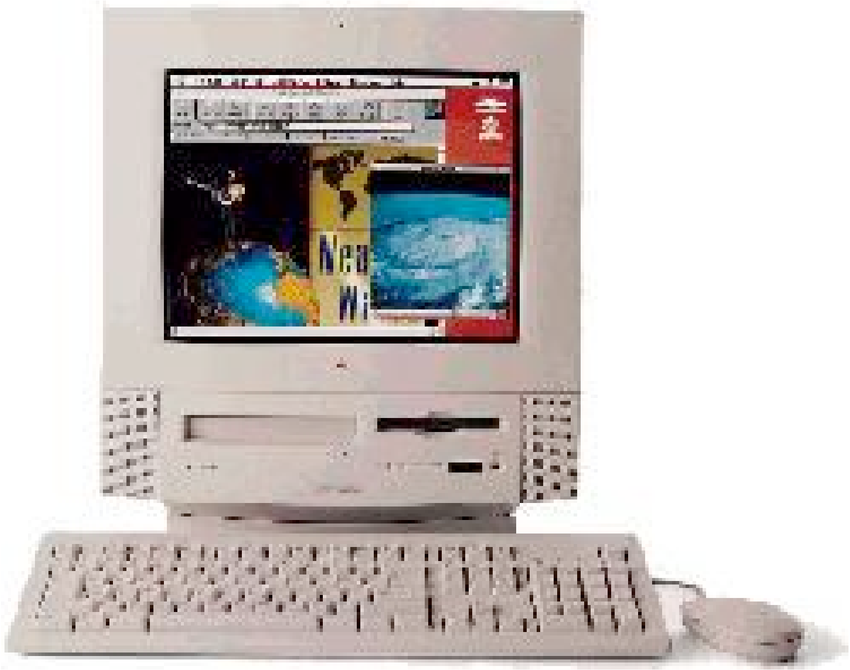
PDP-11: Adressierungsarten

$B_5B_4B_3$	Dezimal	Name	Syntax	Bedeutung
101	5	Autodecrement Indirect	@-(Rn)	Verringere R_n; EA = $[[R_n]]$
111	7	Index Indirect	@X(Rn)	Hole X; Erhöhe PC EA = $[X+[R_n]]$

PDP-11: Adressierungsarten wenn Rn = PC

$B_5B_4B_3$	Dezimal	Name	Syntax	Bedeutung
010	2	Immediate (Autoinkrement)	#N	EA = PC; (d.h., Operand folgt dem Befehl)
011	3	Absolute (Autoinkrement Indirekt)	@#A	EA = [PC]; (d.h., die EA des Operanden folgt dem Befehl)
110	6	Relative (Index)	A	Hole X; Inkrementiere PC; EA = X + [PC]; (d.h. EA ist PC + Offset X; X steht direkt nach dem Befehl)
111	7	Relative Indirekt (Index Indirekt)	@A	Hole X; Inkrementiere PC; EA = [X + [PC]]; (d.h. EA ist Inhalt des [PC] + X; X steht hinter dem Befehl)

Adressierung am Beispiel des PowerPCs



PowerPC: Adressierungsmethoden

- Absolut

  Die Zieladresse ist im Befehl angegeben.
- Register Direkt

  Die Operanden befinden sich in den angegebenen Registern.
- Relativ

  Der Abstand zwischen den Sprungbefehlen und der Zieladresse ist in dem Befehl enthalten.
- Registerindirekt

  Die Zieladresse ist der Inhalt des Link Registers (LR) oder des Count Registers (CTR). Dafür gibt es die Befehle bcctr: branch cond. to count register und bclr: branch cond. to link Register.

PowerPC: Adressierungsmethoden (2)

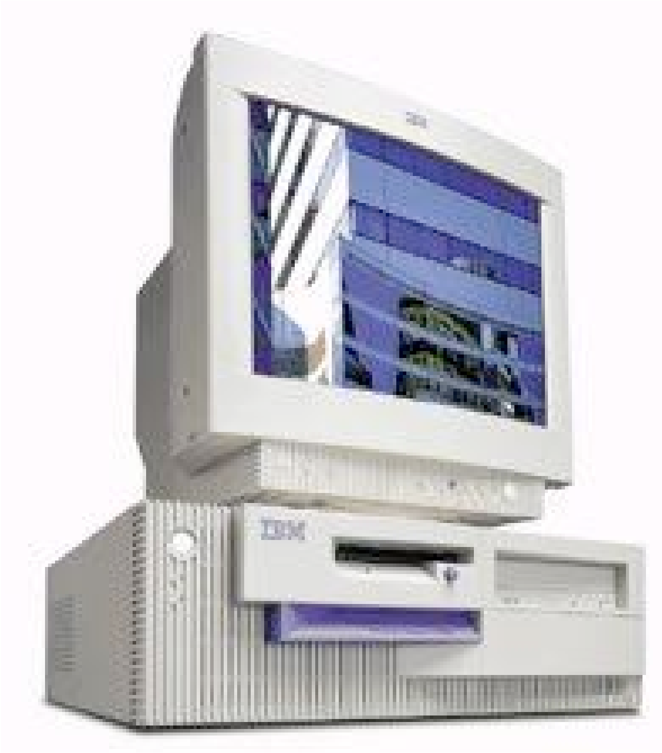
- Indexadressierter Modus
  (Immediate Index Addressing Mode)

  Die effektive Adresse des Operanden ist die Summe der Inhalte des Registers benannt in dem Befehl und einem vorzeichenbehafteten 16 bit Offset, welcher auch in dem Befehl enthalten ist. Die effektive Adresse wird folgendermaßen berechnet: $EA = X + [R_{quelle}]$. Wobei R_{quelle} eines der Allzweckregister R_0 bis R_{31} ist.

- Registerindizierter Adressierungsmodus
  (Register Index Addressing Mode)

  Die effektive Adresse des Operanden ist die Summe der Inhalte der zwei Allzweck Register, die in dem Befehl aufgeführt sind. Die effektive Adresse wird folgendermaßen berechnet: $EA = [R_i] + [R_j]$.

Formate des Pentiums



Pentium: Adressierungsarten

1. Direkt-kodierte Operanden:
   Operand wird mit Befehl übergeben
2. E/A-Port Adressierung:
   Portadresse wird übergeben, von der der Operand zu holen ist
3. Register-Operanden:
   Nummer des Registers wird übergeben, in der sich der Operand befindet
4. Speicher-Operanden:
   Speicher-Adresse wird übergeben, von wo der Operand zu holen ist

Fragen?