8. ReAl-Maschinencodes

Der Maschinencode kann als Bytecode (variable Länge) oder als fest formatierter Befehlscode ausgelegt sein. Die Gestaltung von Maschinenbefehlsformaten ist seit langem bekannt. Deshalb genügt es, einige Beispiele kurz zu beschreiben. Diese Beispiele veranschaulichen:

- Befehle verschiedener Länge (16, 32, 64 Bits sowie Bytecodes mit variabler Länge),
- Befehlsformate mit möglichst großem Adressierungsvermögen (= langen Adreßangaben),
- Befehlsformate, die möglichst viele gleichzeitig auslösbare Wirkungen unterstützen,
- Formate mit kurzen und langen Befehlen,
- Befehlsformate mit flachem und mit geteiltem Ressourcenadreßraum,
- die Nutzung von Halteregistern zum Übergeben von Angaben, die nicht in das jeweilige Befehlsformat passen.

Einige wichtige Gesichtspunkte:

- Möglichst lange Adreßangaben,
- ausreichend lange Ressourcentypangaben im s-Operator (8 Bits sind typischerweise die unterste Grenze),
- einfache Decodierung,
- gute Ausnutzung der Befehlslänge,
- Vorkehrungen zum Eintragen elementarer Direktwerte. Hierfür wird eine zusätzliche Variante des p-Operators vorgesehen (p_imm = p immediate).
- Reserven zum Codieren weiterer Befehle.

Ist die Befehlsliste vorzugsweise zur softwareseitigen Emulation vorgesehen, so kommt es auf lange Adreßangaben an, während die gleichzeitige Auslösung mehrerer Funktionen (Parallelarbeit) praktisch bedeutungslos ist (sie kann vom Emulator ohnehin nicht unterstützt werden). Demgegenüber geht es in Befehlslisten für spezielle Hardware (Spezialprozessoren, Verarbeitungseinrichtungen in FPGAs) vor allem um die Parallelverarbeitung. Die Ressourcenadreßangaben müssen lediglich so lang sein, daß sie die tatsächlich vorhandene Hardware unterstützen können. Für universelle Hardware (Mikrocontroller, Hochleistungsprozessoren) ist typischerweise ein Kompromiß zwischen Adreßlänge und gleichzeitig auslösbaren Funktionen zu finden.

8.1 Beispiel 1: Bytecodes

Befehlsformate mit variabler Länge sind in vielen Rechnerarchitekturen üblich. Ein solcher Befehl beginnt mit einem Operationscodebyte, das sowohl die Befehlswirkung als auch die Anzahl der nachfolgenden Bytes bestimmt. Im folgenden Beispiel (Tabelle 8.1, Abb. 8.1) hat – im Gegensatz zu den maschinenunabhängigen Bytecodes (Kapitel 7) – jeder Befehl nur eine einzige Wirkung (beispielsweise müssen, um 5 gleichartige Ressourcen auszuwählen, 5 s-Operatoren gegeben werden).

Hinweise:

1. Befehlsformate gemäß Tabelle 8.1 haben typischerweise genügend Reserven; das Adressierungsvermögen der einzelnen Angaben wird praktisch nie vollständig ausgenutzt werden.
2. Es werden zweckmäßigerweise (Speicherplatzersparnis) mehrere p_imm-Operatoren mit verschieden langen Direktwerten vorgesehen.

Operator	1. Angabe	2. Angabe	3. Angabe	4. Angabe
s	Ressourcentyp			
s_a	Ressourcentyp	Ressource		
p	Variablenadresse	Ressource	Parameter	
p_imm	Direktwert	Ressource	Parameter	
y	Ressource			
a	Ressource	Variablenadresse		
l	Ressource	Parameter	Ressource	Parameter
c	Ressource	Parameter	Ressource	Parameter
d	Ressource	Parameter	Ressource	Parameter
r	Ressource			

Tabelle 8.1 Befehlsformate im Bytecode (geteilter Ressourcenadreßraum). In Befehlen für einen flachen Ressourcenadreßraum entfallen die Parameterangaben. Zu den Befehlswirkungen vgl. Tabelle 7.2

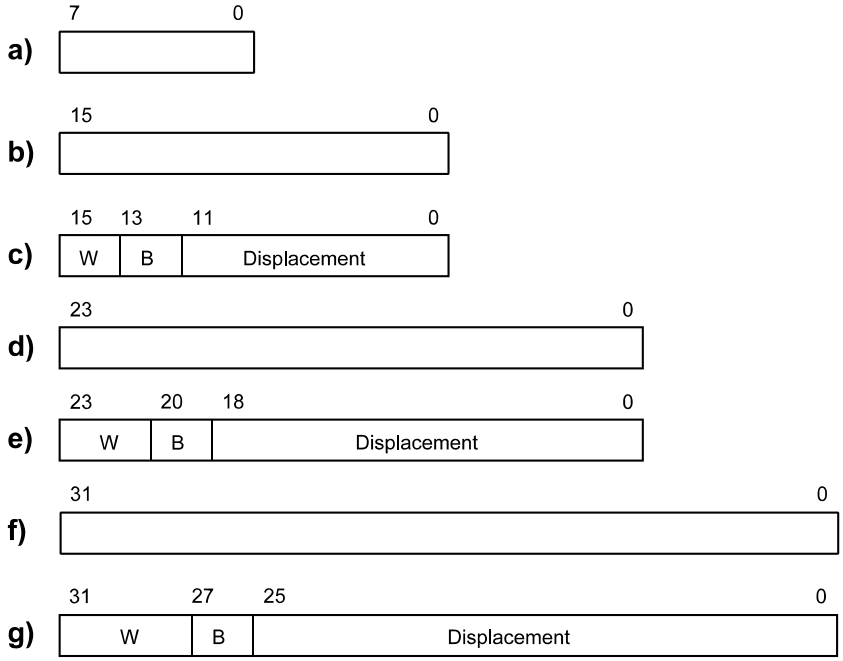

Abb. 8.1 Formate von Bytecodeangaben

a) 1 Byte. Für Operationscodes, Ressourcentypen, Ressourcenadressen, Parameteradressen und Direktwerte.

b) 2 Bytes. Für Ressourcentypen, Ressourcenadressen, Parameteradressen und Direktwerte.

c) Variablenadresse, 2 Bytes. W = Zugriffsbreite, B = Basisadreßregister (vgl. Tabelle 8.3). 4 verschiedene Zugriffsbreiten, 4 Basisadreßregister, 12 Bits Displacement.

d) 3 Bytes. Für Ressourcentypen, Ressourcenadressen, Parameteradressen und Direktwerte.

e) Variablenadresse, 3 Bytes. W = Zugriffsbreite, B = Basisadreßregister (vgl. Tabelle 8.3). 8 verschiedene Zugriffsbreiten, 4 Basisadreßregister, 19 Bits Displacement.

f) 4 Bytes. Für Ressourcentypen, Ressourcenadressen, Parameteradressen und Direktwerte.

g) Variablenadresse, 4 Bytes. W = Zugriffsbreite, B = Basisadreßregister (vgl. Tabelle 8.3). 16 verschiedene Zugriffsbreiten, 4 Basisadreßregister, 26 Bits Displacement.

Die Befehlsformate für eine bestimmte Maschine lassen sich wie aus einem Baukasten zusammenstellen (Tabelle 8.2).

System	Ressourcen-typ	Ressourcen-adresse	Direkt-wert*)	Variablen-adresse	Parameter-adresse
Hardware für Embedded Systems	1	1...2	1, 2	2	1
Hocheistungshardware	1...2	1...2	1, 2, 3, 4	4	1
Software (Emulation) für Embedded Systems	2	2 oder 3	1, 2, 3, 4	2 oder 3	-
Software (Emulation) für den oberen Leistungsbereich	3 oder 4	4	2, 4	3 oder 4	-

*): vgl. den Hinweis 2 am Anfang von Abschnitt 8.1

Tabelle 8.2 Typische Auslegungen von Befehlsformaten im Bytecode

8.2 Beispiel 2: 32 Bits, zwei Wirkungen

Das Beispiel (Abb. 8.2, Tabellen 8.3, 8.4) betrifft ein 32-Bit-Befehlswort und Adreßangaben mittlerer Länge. Jeder Befehl entspricht einem vollständigen Operator. Manche Befehle können zwei Wirkungen auslösen. Der Ressourcenadreßraum umfaßt maximal 4096 Parameter. Die 12 Adreßbits können auch geteilte Ressourcenadressen aufnehmen, z. B. für 1024 Ressuorcen mit 4 Parametern oder 512 Ressourcen mit 8 Parametern. Anwendungsbeispiele: nach ReAl-Prinzipien ausgelegte Hochleistungsprozessoren, Spezialprozessoren, umfangreiche Verarbeitungseinrichtungen in FPGAs usw.

- Befehlslänge: 32 Bits
- Ressourcenadresse: 12 Bits (flacher Adreßraum)
- Ressourcentypangabe: 12 Bits
- Direktwertlänge: 16 Bits
- Variablenadresse (Displacement): 14 Bits
- Festlegung der Zugriffsbreite: im Befehl
- Besonderheiten:
  - der y-Operator kann zwei Ressourcen gleichzeitig aktivieren,
  - der s-Operator kann zwei Ressourcen gleichzeitig auswählen.

Variante: y-Operator mit Funktionscode
Der y-Operator kann nur eine Ressource aktivieren, Die verbleibenden 12 Bits enthalten den zugehörigen Funktionscode (für Ressourcen mit wählbarer Funktion).

W = Zugriffsbreite:	B = Basisadresse:
• 1 Byte	• Stackpointer (SP)
• 2 Bytes	• Base Pointer (BP)
• 4 Bytes	• Global Pointer (GP)
• reserviert	• reserviert

Tabelle 8.3 Codierung von Zugriffsbreite und Basisadresse

Operator	31		29		27		25		23 12	11 0
y	0	0	x	x	0	0	0	x	2. Ressource (12)	1. Ressource (12)
c	0	0	x	x	0	1	0	x	2. Ressource (12)	1. Ressource (12)
d	0	0	x	x	0	1	1	x	2. Ressource (12)	1. Ressource (12)
s	0	0	x	x	1	0	0	x	2. Ressourcentyp (12)	1. Ressourcentyp (12)
r	0	0	x	x	1	0	1	x	2. Ressource (12)	1. Ressource (12)
l	0	0	W		0	0	1	x	2. Ressource (12)	1. Ressource (12)
s_a	0	0	x	x	1	1	1	1	Ressourcenadresse	Ressourcentyp (12)
a	0	1	W		B		Displacement (14)			Ressource (12)
p	1	0	W		B		Displacement (14)			Ressource (12)
p_imm	1	1	W		Immediate (16)					Ressource (12)

Abb. 8.2 ReAl-Maschinencode. 32 Bits mit zwei Wirkungen

Operator	Wirkung
y	Auslösung der Ergebnisberechnung in den angegebenen Ressourcen. Die Adreßangaben betreffen den Funktionscode-Parameter der jeweiligen Ressourcen. Adresse = 0: keine Auslösung
c	Herstellung einer Verkettung 1. Ressource => 2. Ressource. Die Adreßangaben betreffen einen Operanden- und einen Ergebnisparameter
d	Trennung der Verkettung 1. Ressource => 2. Ressource. S. auch unter c.
s	Auswählen (Anfordern) von zwei Ressourcen gemäß Typangabe. Typ = 0: keine Auswahl
r	Freigeben der angegebenen Ressourcen. Die Adreßangaben betreffen den Funktionscode-Parameter der jeweiligen Ressourcen. Adresse = 0: keine Auslösung
l	Parametertransport 1. Ressource => 2. Ressource (Zugriffsbreite W)
s_a	eine Ressource gemäß Typangabe auswählen und die angegebene Ressourcenadresse zuweisen. Die Ressourcenadresse bildet den Anfang für die Ressourcennumerierung in weiteren s-Operatoren (+1-Zählung). Typ = 0: keine Auswahl (nur Einstellen der Anfangsadresse)
a	Ergebnistransport Ressource => Parameteradresse (<Basis B + Displacement>, Zugriffsbreite W)
p	Operandentransport Parameteradresse (<Basis B + Displacement>, Zugriffsbreite W) => Ressource
p_imm	Direktwert-Parametertransport Immediate => Ressource. Direktwert wird gemäß Zugriffsbreite W vorzeichengerecht erweitert bzw. abgeschnitten

Tabelle 8.4 Befehlswirkungen

8.3 Beispiel 3: 32 Bits, lange Adressen

Das Beispiel (Abb. 8.3, Tabellen 8.5 bis 8.7) betrifft ein 32-Bit-Befehlswort und extrem lange Adreßangaben (28 Bits). Anwendung: vor allem für die softwareseitige Emulation (virtuelle ReAl-Maschinen) in oberen Leistungsbereichen. Es ist nicht möglich, zwei Angaben in einem 32-Bit-Wort unterzubringen. Deshalb werden in der Plattform vier Halteregister vorgesehen, die mit u-Operatoren geladen werden können (Tabelle 8.5). Manche Operatoren erfordern deshalb zwei Befehle. Der Ressourcenadreßraum umfaßt maximal 256 M Parameter. Die 28 Adreßbits können auch geteilte Ressourcenadressen aufnehmen, z. B. für 16 M Ressourcen mit 16 Parametern oder für 1 M Ressuorcen mit 256 Parametern.

- Befehlslänge: 32 Bits

- Ressourcenadresse: 28 Bits (flacher Adreßraum)
- Ressourcentypangabe: 26 Bits
- Direktwertlänge: 26 Bits
- Variablenadresse (Displacement): 24 Bits
- Festlegung der Zugriffsbreite: im Befehl
- Zwischenpufferung: 4 Halteregister

Halteregister	geladen mit	genutzt von
1: Variablenadresse	u_va	p
2: Direktwert	u_imm	p_imm
3: Ressourcenadresse	u_rs	l, c, d, a
4: Ressourcenadreßzähler	u_ra	s

Tabelle 8.5 Halteregister

Hinweis:
Die Anordnung von Halteregistern ermöglicht – im Gegensatz zur Verdoppelung der Befehlslänge – oftmals eine Mehrfachnutzung der eingetragenen Angaben:

- Ein Direktwert zu mehreren Ressourcen (p_imm),
- eine Variable zu mehreren Ressourcen (p),
- ein Ergebnis zu mehreren Variablen (a),
- ein Ergebnis zu mehreren Operanden (l, c, d).

Operator	31		29		27	25	23 0
andere	0	0	0	0	*)	res.	
s	0	0	0	0	11	Ressourcentyp (26)	
r	0	0	0	1	Ressource (28)		
p_imm2	0	0	1	0	Ressource (28)		
y	0	0	1	1	Ressource (28)		
l_2	0	1	0	0	Ressource (28)		
c_2	0	1	0	1	Ressource (28)		
d_2	0	1	1	0	Ressource (28)		
u_rs	0	1	1	1	Ressource (28)		
a_2	1	0	0	W	B	Displacement (24)	
u_va	1	0	1	W	B	Displacement (24)	
u_imm	1	1	0	W	Immediate (26)		
p_2	1	1	1	0	Ressource (28)		
u_ra	1	1	1	1	Ressourcenadresse (28)		

*): Codes 00, 01, 10

Abb. 8.3 ReAl-Maschinencode. 32 Bits mit langen Adressen

W = Zugriffsbreite: • 1 Byte • 2 Bytes • 4 Bytes • 8 Bytes • 16 Bytes • reserviert	B = Basisadresse: • Stackpointer (SP) • Base Pointer (BP) • Global Pointer (GP) • reserviert

Tabelle 8.6 Codierung von Zugriffsbreite und Basisadresse

Operator	Wirkung
s	Auswählen (Anfordern) einer Ressourcen gemäß Typangabe. Zuweisung der Ressourcenadresse gemäß Ressourcenadreßzähler (Halteregister 4)
r	Freigeben der angegebenen Ressource
y	Auslösung der Ergebnisberechnung in der angegebenen Ressource
l_2	Parametertransport 1. Ressource => 2. Ressource (Zugriffsbreite W). 1. Parameter gemäß Halteregister 3, 2. Parameter gemäß Adreßangabe. Vollständiger l-Operator: u_rs (1. Ressource); l_2 (2. Ressource)
c_2	Herstellung einer Verkettung 1. Ressource => 2. Ressource. 1. Parameter gemäß Halteregister 3, 2. Parameter gemäß Adreßangabe. Vollständiger c-Operator: u_rs (1. Ressource); c_2 (2. Ressource)
d_2	Trennung der Verkettung 1. Ressource => 2. Ressource. 1. Parameter gemäß Halteregister 3, 2. Parameter gemäß Adreßangabe. Vollständiger d-Operator: u_rs (1. Ressource); d_2 (2. Ressource)
u_rs	Laden der Adreßangabe ins Halteregister 2 (Ressourcenadresse)
a_2	Ergebnistransport Ressource => Variablenadresse (<Basis B + Displacement>, Zugriffsbreite W). Parameter gemäß Halteregister 2. Vollständiger a-Operator: u_rs (Ressource); a_2 (Parameteradresse)
pimm_2	Direktwert-Parametertransport Immediate => Ressource. Direktwert gemäß Halteregister 2. Direktwert wird gemäß Zugriffsbreite W vorzeichengerecht erweitert. Vollständiger p_imm-Operator: u_imm (Direktwert); p_imm2 (Ressource)
u_va	Laden der Variablenadresse ins Halteregister 1
u_imm	Laden des Direktwertes ins Halteregister 2
p_2	Operandentransport Variablenadresse (<Basis B + Displacement>, Zugriffsbreite W) => Ressource. Variablenadresse gemäß Halteregister 1. Vollständiger p-Operator: u_va (Variablenadresse); p_2 (Ressource)
u_ra	Ressourcenadresse für s-Operator einstellen (Halteregister 4). Die Ressourcenadresse bildet den Anfang für die Ressourcennumerierung in nachfolgenden s-Operatoren

Tabelle 8.7 Befehlswirkungen

8.4 Beispiel 4: 16 Bits, zwei Wirkungen

Es handelt sich um eine Abwandlung des Beispiels 2 mit kurzen Befehlen (16 Bits) und zwei Ressourcenangaben im Befehl (Abb. 8.4, Tabelle 8.8). Die einzelne Ressourcenadresse ist auf 6 Bits beschränkt. Hiermit können z. B. 16 Ressourcen mit jeweils maximal 4 Parametern oder 8 Ressourcen mit jeweils 8 Parametern unterstützt werden. Anwendungsbeispiele: nach ReAl-Prinzipien ausgelegte Mikrocontroller, kleinere Spezialprozessoren, Verarbeitungseinrichtungen in FPGAs usw. Bei Beschränkung auf 16 Bits werden die Displacementangaben der Variablenadressen und die Direktwerte sehr kurz. Deshalb gibt es von den p-, p_imm- und a-Operatoren jeweils eine Langversion, die das nachfolgende 16-Bit-Wort einschließt.

- Befehlslänge: 16 bzw. 32 Bits
- Ressourcenadresse: 6 Bits (flacher Adreßraum)
- Ressourcentypangabe: 12 Bits
- Direktwertlänge: 6 oder 20 Bits
- Variablenadresse (Displacement): 4 oder 18 Bits
- Festlegung der Zugriffsbreite: in den Ressourcen (16-Bit-Befehle) oder im Befehl (32-Bit-Befehle)
- Besonderheit: der y-Operator kann zwei Ressourcen gleichzeitig aktivieren.

Operator	15 12	11 6	5 0	15 **2. 16-Bit-Wort** 0
p_l	0 0 1 0	Ressource	B Displace-ment	W Displacement
p_s	0 0 1 1	Ressource	Displacement	
p_imml	0 1 0 0	Ressource	Immediate	W Immediate (14)
p_imms	0 1 0 1	Ressource	Immediate	
a_l	0 1 1 0	Ressource	B Displace-ment	W Displacement
a_s	0 1 1 1	Ressource	Displacement	
l	1 0 0 0	Ressource	Ressource	
c	1 0 0 1	Ressource	Ressource	
d	1 0 1 0	Ressource	Ressource	
r	1 0 1 1	Ressource	Ressource	
y	1 1 0 0	Ressource	Ressource	
s	1 1 0 1	Ressourcentyp		
s_a	1 1 1 1	Ressourcentyp		Ressource

Abb. 8.4 ReAl-Maschinencode. 16 Bits mit zwei Wirkungen. Zu den Angaben W und B vgl. Tabelle 8.3

Operator	**Wirkung**
p_l	Operandentransport Parameteradresse (<Basis B + Displacement>, Zugriffsbreite W) => Ressource. Lange Displacementadresse (18 Bits)
p_s	Operandentransport Parameteradresse (<Basis B + Displacement>, Zugriffsbreite W) => Ressource. Kurze Displacementadresse (4 Bits)
p_imml	Direktwert-Parametertransport Immediate => Ressource. Langer Direktwert (20 Bits). Direktwert wird gemäß Zugriffsbreite W vorzeichengerecht erweitert
p_imms	Direktwert-Parametertransport Immediate => Ressource. Kurzer Direktwert (6 Bits). Direktwert wird gemäß Zugriffsbreite W vorzeichengerecht erweitert
a_l	Ergebnistransport Ressource => Parameteradresse (<Basis B + Displacement>, Zugriffsbreite W). Lange Displacementadresse (18 Bits)
a_s	Ergebnistransport Ressource => Parameteradresse (<Basis B + Displacement>, Zugriffsbreite W). Kurze Displacementadresse (4 Bits)
l	Parametertransport 1. Ressource => 2. Ressource (Zugriffsbreite W)
c	Herstellung einer Verkettung 1. Ressource => 2. Ressource. Die Adreßangaben betreffen einen Operanden- und einen Ergebnisparameter

Operator	Wirkung
d	Trennung der Verkettung 1. Ressource => 2. Ressource. S. auch unter c.
r	Freigeben der angegebenen Ressourcen. Die Adreßangaben betreffen den Funktionscode-Parameter der jeweiligen Ressourcen. Adresse = 0: keine Auslösung
y	Auslösung der Ergebnisberechnung in den angegebenen Ressourcen. Die Adreßangaben betreffen den Funktionscode-Parameter der jeweiligen Ressourcen. Adresse = 0: keine Auslösung
s	Auswählen (Anfordern) einer Ressource gemäß Typangabe
c	Herstellung einer Verkettung 1. Ressource => 2. Ressource. Die Adreßangaben betreffen einen Operanden- und einen Ergebnisparameter
d	Trennung der Verkettung 1. Ressource => 2. Ressource. S. auch unter c
s_a	eine Ressource gemäß Typangabe auswählen und die angegebene Ressourcenadresse zuweisen. Die Ressourcenadresse bildet den Anfang für die Ressourcennumerierung in weiteren s-Operatoren (+1-Zählung). Typ = 0: keine Auswahl (nur Einstellen der Anfangsadresse)

Tabelle 8.8 Beispiel 3: Befehlswirkungen

8.5 Beispiel 5: 16 Bits, längere Adressen

Es handelt sich um eine Abwandlung des Beispiels 3 mit kurzen Befehlen (16 Bits) und entsprechend kürzeren, aber durchaus brauchbare Adreßangaben (Abb. 8.5). Mit einer Ressourcenadresse von 12 Bits können z. B. 512 Ressourcen mit jeweils maximal 8 Parametern unterstützt werden. Es ist nicht möglich, zwei Adreßangaben in einem Befehl unterzubringen. Deshalb werden in der Plattform vier Halteregister vorgesehen, die mit u-Operatoren geladen werden können (vgl. Tabelle 8.5). Manche Operatoren erfordern deshalb zwei Befehle. Anwendungsbeispiele: softwareseitige Emulation (virtuelle ReAl-Maschinen), vorzugsweise im Bereich der Embedded Systems, nach ReAl-Prinzipien ausgelegte Prozessoren usw.

- Befehlslänge: 16 Bits
- Ressourcenadresse: 12 Bits (flacher Adreßraum)
- Ressourcentypangabe: 10 Bits
- Direktwertlänge: 11...13 Bits
- Variablenadresse (Displacement): 9 Bits
- Festlegung der Zugriffsbreite: im Operator
- Zwischenpufferung: 4 Halteregister mit 12 Bits

Operator	15		13		11	9	8	0
andere	0	0	0	0	*)		res.	
s					11		Ressourcentyp (10)	
r	0	0	0	1	Ressource (12)			
p_imm2	0	0	1	0	Ressource (12)			
y	0	0	1	1	Ressource (12)			
l_2	0	1	0	0	Ressource (12)			
c_2	0	1	0	1	Ressource (12)			
d_2	0	1	1	0	Ressource (12)			
u_rs	0	1	1	1	Ressource (12)			
a_2	1	0	0	W	B	Displacement (9)		
u_va	1	0	1	W	B	Displacement (9)		

Operator	15		13		11 9	8 0
u_imm	1	1	0	W		Immediate (11)
p_2	1	1	1	0		Ressource (12)
u_ra	1	1	1	1		Ressourcenadresse (12)

*): Codes 00, 01, 10

Abb. 8.5 ReAl-Maschinencode. 16 Bits mit längeren Adressen. S. weiterhin die Tabellen 8.5 bis 8.7

8.6 Beispiel 6: kurze Befehle mit parallel auslösbaren Funktionen

Dieses Beispiel betrifft kurze Befehle, in denen möglichst viele parallel ausführbare Funktionen codiert sind (Abb. 8.6, 8.7, Tabelle 8.9). Es werden bis zu 64 Ressourcen mit jeweils maximal 8 Parametern unterstützt (geteilter Adreßraum). Die Angaben für die einzelnen Operatoren werden abschnittsweise zugeführt. Um die endgültigen Angaben bereitzustellen, werden in der Plattform zwei Halteregister angeordnet. Zum Laden der Halteregister sind zusätzliche Operatoren u_rs1, u_rs2 vorgesehen. Der y-Operator kann bis zu acht Ressourcen gleichzeitig aktivieren (über Bitmaske). Anwendungsbeispiele: nach ReAl-Prinzipien ausgelegte Mikrocontroller, universelle Prozessoren, Spezialprozessoren, Verarbeitungseinrichtungen in FPGAs usw. Dieses Befehlsformat unterstützt vor allem die Parallelarbeit zwischen den Ressourcen (gleichzeitige Operationen, Parametertransporte usw.).

- Befehlslänge: 16 Bits
- Ressourcenadresse: 6 Bits (geteilter Adreßraum)
- Parameteradresse: 3 Bits
- Ressourcentypangabe: 10 Bits
- Direktwertlänge: 9 Bits
- Variablenadresse (Displacement): 9 Bits
- Festlegung der Zugriffsbreite: in den Ressourcen
- gleichzeitig aulösbar sind:
  - 2 Parametertransporte zwischen den Ressourcen
    oder
  - 2 Verkettungssteuerfunktionen
    oder
  - die Aktivierung von maximal 8 Ressourcen
    oder
  - das Eintragen von 2 Ressourcenadressen in ein Halteregister

	11 6	5 0
Halteregister 1	1. Quellressource	1. Zielressource
Halteregister 2	2. Quellressource	2. Zielressource

Abb. 8.6 Belegung der Halteregister

Operator	15 12				11 3								2 0
p	1	0	B		Displacement								Parameter
a	1	1	B		Displacement								Parameter
y	0	0	0	0	0	8.	7.	6.	5.	4.	3.	2.	1.	Ress.sel.
andere	0	0	0	0	1	0	0	res.						
s	0	0	0	0	1	0	1	Ressourcentyp (9)						
u_ra	0	0	0	0	1	1	Ressourcenadresse (6)							

Operator	15			12	11			3	2	0
l	0	0	0	1	1. Quellp.		1. Zielp.		2. Quellp.	2. Zielp.
c	0	0	1	0	1. Quellp.		1. Zielp.		2. Quellp.	2. Zielp.
d	0	0	1	1	1. Quellp.		1. Zielp.		2. Quellp.	2. Zielp.
r	0	1	0	0	1. Ressource				2. Ressource	
u_rs1	0	1	0	1	1. Quellressource				1. Zielressource	
u_rs2	0	1	1	0	2. Quellressource				2. Zielressource	
p_imm	0	1	1	1	Immediate (9)					Parameter

Abb. 8.7 ReAl-Maschinencode. Kurze Befehle mit parallel auslösbaren Funktionen. Zur Basisadreß-angabe B vgl. Tabelle 8.3

Operator	Wirkung
p	Operandentransport Parameteradresse (<Basis B + Displacement>) => Parameter in Ressource. Die Ressourcenadresse befindet sich im 1. Halteregister (1. Zielressource)
a	Ergebnistransport Ressource => Parameteradresse (<Basis B + Displacement>). Die Ressourcenadresse befindet sich im 1. Halteregister (1. Quellressource)
y	Auslösung der Ergebnisberechnung in den angegebenen Ressourcen. Das Feld *Ress.sel.* wählt aus den ingesamt 64 unterstützen Ressourcen eine Gruppe von 8 aus, die über Einzelbits in beliebiger Kombination aktiviert werden können
s	Auswählen (Anfordern) einer Ressource gemäß Typangabe
u_ra	Ressourcenadressen für s-Operator einstellen. Die Ressourcenadresse bildet den Anfang für die Ressourcennumerierung in weiteren s-Operatoren (+1-Zählung)
l	Parametertransporte 1. Quellparameter => 1. Zielparameter, 2. Quellparameter => 2. Zielparameter. Ressourcenadressen für den ersten Transport in Halteregister 1, für den zweiten Transport in Halteregister 2. Quellparameter = 0: kein Transport
c	Herstellung zweier Verkettungen 1. Quellparameter => 1. Zielparameter, 2. Quellparameter => 2. Zielparameter. Ressourcenadressen für die erste Verkettung in Halteregister 1, für die zweite Verkettung in Halteregister 2. Quellparameter = 0: keine Verkettung
d	Trennung zweier Verkettungen 1. Quellparameter => 1. Zielparameter, 2. Quellparameter => 2. Zielparameter. Ressourcenadressen für die erste Verkettung in Halteregister 1, für die zweite Verkettung in Halteregister 2. Quellparameter = 0: keine Wirkung
r	Freigeben der angegebenen zwei Ressourcen
u_rs1	Laden des Halteregisters 1 mit zwei Ressourcenadressen
u_rs2	Laden des Halteregisters 2 mit zwei Ressourcenadressen
p_imm	Direktwert-Parametertransport Immediate => Parameteradresse. Die Ressourcenadresse befindet sich im 1. Halteregister (1. Zielressource)

Tabelle 8.9 Beispiel 5: Befehlswirkungen

8.7 Beispiel 7: 64 Bits, viele parallel auslösbare Funktionen

In diesem Beispiel (Abb.n 8.8...8.11, Tabellen 8.10, 8.11) geht es darum, sehr viele parallel ausführbare Funktionen zu codieren. Hierzu sind längere Befehle erforderlich (64 Bits). Wie in Beispiel 6 werden bis zu 64 Ressourcen mit jeweils maximal 8 Parametern unterstützt (geteilter Adreßraum). Die Angaben für die einzelnen Operatoren werden abschnittsweise zugeführt. Um die endgültigen Angaben bereitzustellen, werden in der Plattform zwei Halteregister angeordnet. Zum Laden der Halteregister sind

zusätzliche Operatoren u_rs1, u_rs2 vorgesehen. Der y-Operator kann bis zu 60 Ressourcen gleichzeitig aktivieren (über Bitmaske). Anwendungsbeispiele: Spezialprozessoren, Verarbeitungseinrichtungen in FPGAs usw.

- Befehlslänge: 64 Bits
- Ressourcenadresse: 6 Bits (geteilter Adreßraum)
- Parameteradresse: 3 Bits
- Ressourcentypangabe: 10 Bits
- Direktwertlänge: 12 Bits
- Variablenadresse (Displacement): 10 Bits
- Festlegung der Zugriffsbreite: in den Ressourcen
- gleichzeitig auslösbar sind:
  - 10 Parametertransporte zwischen den Ressourcen
    oder
  - 10 Verkettungsteuerfunktionen
    oder
  - 4 Parametertransporte zwischen Ressourcen und Plattform
    oder
  - die Aktivierung von maximal 60 Ressourcen
    oder
  - das Eintragen von 10 Ressourcenadressen in ein Halteregister

63 60	59 0
Opcode	Adreß-, Direktwert- oder Steuerangaben

Abb. 8.8 ReAl-Maschinencode. 64 Bits mit vielen parallel aulösbaren Funktionen. Das grundsätzliche Befehlsformat

Opcode	Format	Opcode	Format	Opcode	Format	Opcode	Format
0	y_1	4	res.	8	l	C	r
1	y_2	5	res.	9	p	D	s
2	u_ra	6	c	A	a	E	u_rs2
3	andere	7	d	B	p_imm	F	u_rs1

Abb. 8.9 Operationscodes

Operator	Belegung der Bits 59...0									
l, c, d	s/d1	s/d2	s/d3	s/d4	s/d5	s/d6	s/d7	s/d8	s/d9	s/d10
r	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10
u_1	s1	d1	s2	d2	s3	d3	s4	d4	s5	d5
u_2	s6	d6	s7	d7	s8	d8	s9	d9	s10	d10
y	60 Einzelbits									
p, p_imm,a	parm1			parm2			parm3			parm4
s	rt1		rt2		rt3		rt4		rt5	rt6
u_ra	ra 1 (6)		ra 2 (6)		ra 3 (6)		ra 4 (6)		ra 5 (6)	ra 6 (6)

Abb. 8.10 Übersicht über die einzelnen Befehlsformate. S. weiterhin Tabelle 8.9

Operation	14 12	11 3	2 0
p, a	B	Displacement (10)	Parameter
pimm	Immediate (12)		Parameter

Abb. 8.11 Parameterangabe in den Operatoren p, pimm und a. Zur Basisadreßangabe B vgl. Tabelle 8.3

Halteregister 1	s1	d1	s2	d2	s3	d3	s4	d4	s5	d5
Halteregister 2	s6	d6	s7	d7	s8	d8	s9	d9	s10	d10

Abb. 8.12 Belegung der Halteregister

Operator	**Belegung der Befehlsbits 59...0**
p	4 Parameterangaben (parm1...4) zu 15 Bits; vgl. Abb. 8.10
a	4 Parameterangaben (parm1...4) zu 15 Bits; vgl. Abb. 8.10
y_1	60 Einzelbits zur Aktivierung der Ressourcen 59...0
y_2	60 Einzelbits zur Aktivierung der Ressourcen 63...60 und 55...0
s	6 Ressourcentypangaben (rt1...10) zu 10 Bits
l, c, d	10 Parameteradreßpaare (s/d1...s/d10) (zu 6 Bits. s = Quellparameter (3 Bits), d = Zielparameter (3 Bits)
r	10 Ressourcenadresen (r1...r10) zu 6 Bits
u_rs1	Laden des Halteregisters 1 mit 5 Ressourcenadreßpaaren (s1, d1...s5, d5) zu 12 Bits; vgl. Abb. 8.11
u_rs2	Laden des Halteregisters 2 mit 5 Ressourcenadreßpaaren (s6, d6...s10, d10) zu 12 Bits; vgl. Abb. 8.11
p_imm	4 Parameterangaben (parm1...4) zu 15 Bits; vgl. Abb. 8.10

Tabelle 8.10 Inhalte der Befehlsformate

Operator	**Wirkung**
p	4 Operandentransporte Parameteradresse (<Basis B + Displacement>) => Parameter in Ressource. Die Ressourcenadressen befinden sich im 1. Halteregister (Zielressourcen d1...d4). Parameter = 7: kein Transport
a	4 Ergebnistransporte Ressource => Parameteradresse (<Basis B + Displacement>). Die Ressourcenadressen befinden sich im 1. Halteregister (Quelllressourcen s1...s4). Parameter = 7: kein Transport
y_1	Auslösung der Ergebnisberechnung in den ersten 60 Ressourcen (59...0).
y_2	Auslösung der Ergebnisberechnung in den letzten 4 und den ersten 56 Ressourcen (63...60, 55...0).
s	Auswählen (Anfordern) von 6 Ressourcen gemäß Typangabe (rt1...6). Ressourcentyp = 0. keine Auswahl
u_ra	Ressourcenadressen für s-Operator einstellen. Die 6. Ressourcenadresse bildet den Anfang für die Ressourcennumerierung in weiteren s-Operatoren (+1-Zählung)
l	10 Parametertransporte Quellparameter => Zielparameter (s1 => d1...s10 => d10). Ressourcenadressen für die ersten 5 Transportvorgänge (1...5) in Halteregister 1, für die zweiten 5 Transportvorgänge(6...10) in Halteregister 2. Quellparameter = 0: kein Transport

Operator	Wirkung
c	Herstellen von 10 Verkettungen Quellparameter => Zielparameter (s1 => d1...s10 => d10). Ressourcenadressen für die ersten 5 Verkettungen (1...5) in Halteregister 1, für die zweiten 5 Verkettungen (6...10) in Halteregister 2. Quellparameter = 0: keine Verkettung
d	Trennung von 10 Verkettungen Quellparameter => Zielparameter (s1 => d1...s10 => d10). Ressourcenadressen für die ersten 5 Verkettungen (1...5) in Halteregister 1, für die zweiten 5 Verkettungen (6...10) in Halteregister 2. Quellparameter = 0: keine Wirkung
r	Freigeben der angegebenen 10 Ressourcen (r1...r10)
u_rs1	Laden des Halteregisters 1 mit 10 Ressourcenadressen zu 6 Bits (s1, d1...s5, d5)
u_rs2	Laden des Halteregisters 2 mit 10 Ressourcenadressen zu 6 Bits (s6, d6...s10, d10)
p_imm	4 Direktwert-Parametertransporte Immediate => Parameteradresse. Die Ressourcenadressen befinden sich im 1. Halteregister (Zielressourcen d1...d4). Parameter = 7: kein Transport

Tabelle 8.11 Beispiel 6: Befehlswirkungen