8. Machine Codes

Machine codes can be byte codes (variable length) or fixed-format instruction codes. The principal layout of these machine instruction formats is well known in computer architecture. Therefore, it is sufficient to briefly explain a few examples. The following examples illustrate:

- Instructions of different length (16, 32, 64 bits as well as byte codes with variable length).
- Instruction formats with an addressing capability as large as possible (in other words, with long address fields).
- Instruction formats that can initiate simultaneously as many as possible functions.
- Formats with short and long instructions.
- Instruction formats with flat and split resource address space.
- Utilization of buffer registers for transfer of data that does not fit the respective instruction format.

Some important considerations of the instruction format design:

1. Address fields should be as long as possible.
2. Resource type fields (in the s-operator) should be sufficiently long (8 bits are typically the lowest limit).
3. Instruction decoding should be done with comparatively simple means.
4. The bits of the instruction format should be used efficiently.
5. It should be straightforward, to deliver elementary immediate values as parameters[1)].
6. In order to be able to encode further instructions, sufficient reserves should be provided.

When the instruction set is preferably provided for software-based emulation, long address fields are important while the simultaneous initiation of several functions (parallel operation) is practically meaningless (it cannot be supported by the emulator anyway). In contrast to this, in the instruction sets for special hardware (special processors, processing circuitry in FPGAs) parallel processing is the primary concern. The resource address information must be only of such a length that the actually present hardware can be supported. For general-purpose hardware (microcontrollers, high-performance processors) typically a compromise between address length and the support of parallel operation functions can be found.

All examples contain unused bit positions or special reserved formats that can be used for extension of the respective instruction set (for example, for m-operators, h-operators, u-operators; for s-operators for requesting resources via the Internet; for instructions controlling the platform etc.). Additional instructions can also occupy more than one instruction word.

---

1): For this purpose, an additional variant of the p-operator is provided (p_imm = p immediate; see also table 7.1).

8.1 Example 1: Byte Codes

The instructions are comprised of sequential bytes (byte code). Instruction formats with variable length can be found in many computer architectures. Such an instruction begins with an operation code byte that determines the instruction function as well as the number and meaning of subsequent bytes. Those bytes constitute information fields containing ordinal numbers, addresses or immediate values. In the example (table 8.1 and fig. 8.1), each instruction has only a single function[1)] (for example, five s-operators must be provided in order to select five identical resources).

operator	1st field	2nd field	3rd field	4th field
s	resource type	–	–	–
s_a	resource type	resource address	–	–
p	variable address[1)]	resource address	parameter address	–
p_imm	immediate value	resource address	parameter address	–
y	resource address	–	–	–
a	resource address	variable address[1)]	–	–
l	resource address	parameter address	resource address	parameter address
c	resource address	parameter address	resource address	parameter address
d	resource address	parameter address	resource address	parameter address
r	resource address	–	–	–

1): Displacement to address variables in memory.

Table 8.1 An overview over the instruction formats of the example byte code. Each of the fields consists of one or more bytes (refer to fig. 8.1).

Table 8.1 provides an overview of the instruction formats for a split resource address space. In the instructions for a flat resource address space the parameter address fields are not needed. The functions of the instructions have been explained in table 7.1.

The fields according to table 8.1 and fig. 105 can be used as some kind of modular system from which the instruction formats for a certain machine can be combined in a building block fashion. Table 8.2 contains the length of the individual fields (in bytes) for several typical areas of applications. Those instruction formats have enough reserves. The addressing capability of the individual fields is practically never completely utilized.

1): This is in contrast to the byte codes discussed in chapter 7.

example	resource type	resource address	immediate value[1)]	variable address	parameter address
hardware for embedded systems	1	1 or 2	1, 2	2	1
high-performance hardware	1 or 2	1 or 2	1, 2, 3, 4	4	1
software (emulation) for embedded systems	2	2 or 3	1, 2, 3, 4	2 or 3	-
software (emulation) for high-performance systems	3 or 4	4	2, 4	3 or 4	-

1:) In order to reduce code size, it is advantageous to provide several p_imm operators having immediate values of differently length.

Table 8.2 Typical sizes of the particular fields (in bytes) for different areas of application.

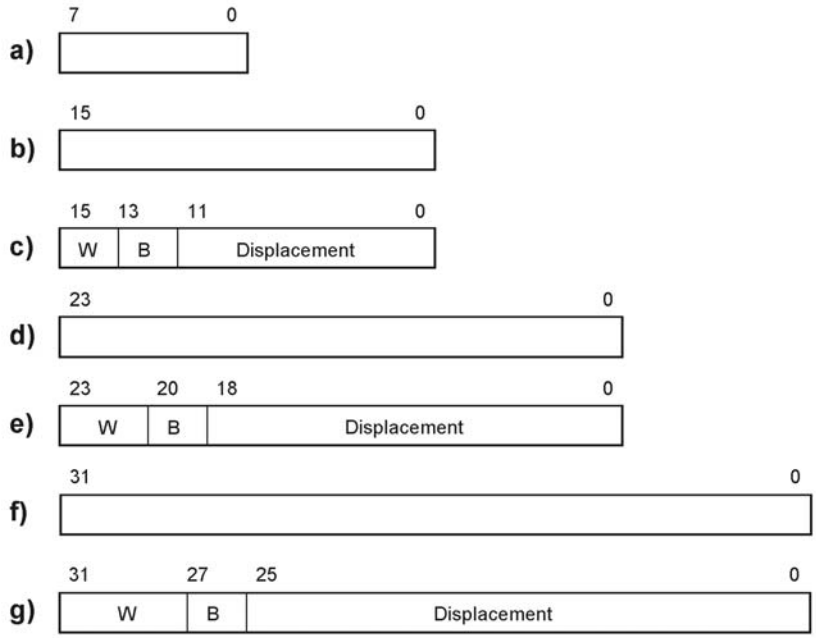

Fig. 8.1 Different formats of byte code instruction fields.

Fig. 8.1 shows the formats of the fields of which the instructions are comprised:

a) 1 byte. For operation codes, resource types, resource addresses, parameter addresses, and immediate values.
b) 2 bytes. For operation codes, resource types, resource addresses, parameter addresses, and immediate values.
c) Variable address (in memory), 2 bytes. W = access width, B = base address register (refer to table 8.4); 4 different access widths, 4 base address registers, 12 bits displacement.
d) 3 bytes. For operation codes, resource types, resource addresses, parameter addresses, and immediate values.
e) Variable address (in memory), 3 bytes. W = access width, B = base address register (refer to table 8.4); 8 different access widths, 4 base address registers, 19 bits displacement.

f) 4 bytes. For operation codes, resource types, resource addresses, parameter addresses, and immediate values.
g) Variable address (in memory), 4 bytes. W = access width, B = base address register (refer to table 8.4); 16 different access widths, 4 base address registers, 26 bits displacement.

8.2 Example 2: 32 bits, two functions

The instruction words contain address fields of intermediate length. Each instruction corresponds to a complete operator. Some instructions can initiate two functions. The resource address space comprises up to 4,096 parameters. The 12 resource address bits can also contain split resource addresses, for example, for 1,024 resources with four parameters or 512 resources with 8 parameters. Applications: high-performance general-purpose processors, special processors, complex processing devices in FPGAs and the like. Fig. 8.2 provides an overview of the machine code, table 8.3 describes the instruction functions. Table 8.4 shows how the access width W and the base address B are encoded.

Overview:

- Instruction length: 32 bits
- Resource address: 12 bits (flat address space)
- Resource type: 12 bits
- Immediate value: 16 bits
- Variable address (displacement): 14 bits, up to 4 base address registers (B)
- Encoding of access width: in the instruction; up to 4 variants (W)
- Special features:
  - the y-operator can activate two resources at the same time
  - the s-operator can select two resources at the same time

operator	31	29	27	25	23 12	11 0
y	0 0	x x	0 0	0 0	2nd resource (12)	1st resource (12)
y_f	0 0	x x	0 0	0 1	function code (12)	resource (12)
c	0 0	x x	0 1	0 x	2nd resource (12)	1st resource (12)
d	0 0	x x	0 1	1 x	2nd resource (12)	1st resource (12)
s	0 0	x x	1 0	0 x	2nd resource type (12)	1st resource type (12)
r	0 0	x x	1 0	1 x	2nd resource (12)	1st resource (12)
l	0 0	W	0 0	1 x	2nd resource (12)	1st resource (12)
s_a	0 0	x x	1 1	1 1	resource address (12)	resource type (12)
a	0 1	W	B	displacement (14)		resource (12)
p	1 0	W	B	displacement (14)		resource (12)
p_imm	1 1	W	immediate (16)			resource (12)

Fig. 8.2 Instruction formats.

operator	function
y	initiate the information processing operations of the selected resources (yield). The fields address the function code parameters of the respective resources. Value = 0: no initiation
y_f	initiate the information processing operations in the selected resource according to the selected function code. Function code = 0: no initiation. Application: for resources with selectable functions
c	establish a concatenation 1st resource => 2nd resource. The fields address an operand parameter and a result parameter
d	disconnect the concatenation 1st resource => 2nd resource; see also under c
s	select up to two resources out of the resource pool according to the type fields. Value = 0: no resource selected
r	return the selected resources to the resource pool. The fields address the function code parameters of the respective resources. Value = 0: no effect
l	move data between the selected resources (link). 1st resource => 2nd resource. Current access width encoded in W
s_a	select (request out of the resource pool) a resource according to the type field and assign the selected resource address. This resource address is the base of resource enumeration in subsequent s-operators (+1 increment). Type = 0: no resource selected (but a resource address other than zero will be assigned)
a	store result from the selected resource into system memory (assign result). Resource => variable address in memory = <base B + displacement>. Current access width encoded in W
p	fetch parameter out of system memory and move it into the selected resource (parameter passing). Content of variable address in memory = <base B + displacement> => resource. Current access width encoded in W
p_imm	load immediate value into the selected resource (parameter passing). If access width W is greater than 16 bits, the immediate value will be sign-extended

Table 8.3 Instruction overview.

W = access width: • 1 byte • 2 bytes • 4 bytes • reserved	B = base address: • stack pointer (SP • base pointer (BP) • global pointer (GP) • reserved

Table 8.4 Encoding of access width (W) and base address register selection (B).

8.3 Example 3: 32 bits, long addresses

In contrast to example 2, the address fields of those 32 bit instructions are of 28 bit length. Application: primarily for software emulation (virtual machines) in the upper performance range. It is not possible to accommodate two address or data fields in a 32 bit word. To supply all the necessary parameters, in the platform four buffer registers are provided that can be loaded with u-operators. Some operators require therefore two instructions. The resource address space comprises maximally 256M parameters. The 28 address bits can also accommodate split resource addresses, for example, for 16M resources with 16 parameters or for 1M resources with 256 parameters. Table 8.5 shows how the buffer registers are used. Fig. 8.3 provides an overview of the machine code, table 8.8 describes the instruction functions. In table 8.9 it is shown how the access width W and the base address B are encoded.

Overview:

- Instruction length: 32 bits
- Resource address: 28 bits (flat address space)
- Resource type: 26 bits
- Immediate value: 26 bits
- Variable address (displacement): 24 bits; up to 4 base address registers (B)
- Encoding of access width: in the instruction; up to 8 variants (W)
- Buffering: four buffer registers

buffer register	loaded with	utilized by
1: address of variable	u_va	p
2: immediate value	u_imm	p_imm
3: resource address	u_rs	l, c, d, a
4: resource address counter	u_ra	s

Table 8.5 Buffer registers.

The arrangement of buffer registers enables, in contrast to the self-evident doubling of the instruction length, often multiple utilization of the entered information:

- Transport of immediate value to several resources (p_imm).
- Transport of a variable to several resources (p).
- Assignment of a result to several variables (a).
- Transport of a result to several operands (l, c, d).

operator	31		29		27	25	23 0
others (reserved)	0	0	0	0	*)		res.
s	0	0	0	0	11		resource type (26)
r	0	0	0	1			resource (28)
p_imm2	0	0	1	0			resource (28)
y	0	0	1	1			resource (28)
l_2	0	1	0	0			resource (28)
c_2	0	1	0	1			resource (28)
d_2	0	1	1	0			resource (28)
u_rs	0	1	1	1			resource (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			resource (28)
u_ra	1	1	1	1			resource address (28)

*): Codes 00, 01, 10

Fig. 8.3 Instruction formats.

operator	function
s	select a resource out of the resource pool according to the type field. Assignment of the resource address according to resource address counter (buffer register 4). Value = 0: no resource selected
r	return teh selected resource to the resource pool. The field addresses the function code parameter of the respective resource. Value = 0: no effect
y	initiate the information processing operations of the selected resources(yield). The field addresses the function code parameter of the respective resources. Value = 0: no initiation
l_2	move data between the selected resources (link). 1st resource => 2nd resource. 1st parameter according to buffer register 3, 2nd parameter according to address field. complete l-operator: u_rs (1st resource); l_2 (2nd resource). Current access width encoded in W
c_2	establish a concatenation 1st resource => 2nd resource. 1st parameter according to buffer register 3, 2nd parameter according to address field. Complete c-operator: u_rs (1st resource); c_2 (2nd resource)
d_2	disconnect the concatenation 1st resource => 2nd resource;. 1st parameter according to buffer register 3, 2nd parameter according to address field. Complete d-operator: u_rs (1st resource); d_2 (2nd resource)
u_rs	load the content of the address field into buffer register 3 (resource address)

operator	function
a_2	store result from the selected resource into system memory (assign result). Resource => variable address in memory = <base B + displacement>. Parameter according to buffer register 3. Complete a-operator: u_rs (resource); a_2 (parameter address). Current access width encoded in W
pimm_2	load immediate value into the selected resource (parameter passing). Immediate value according to buffer register 2. Complete p_imm operator: u_imm (immediate value); p_imm2 (resource). If access width W is greater than 16 bits, the immediate value will be sign-extended
u_va	load the variable address into buffer register 1
u_imm	load the immediate value into buffer register 2
p_2	fetch parameter out of system memory and move it into the selected resource (parameter passing). Content of variable address in memory = <base B + displacement> => resource. Address of the variable according to buffer register 1. Complete p-operator: u_va (address of variable); p_2 (resource). Current access width encoded in W
u_ra	load resource address for s-operator into buffer register 4. This resource address is the base of resource enumeration in subsequent s-operators (+1 increment). Complete s_a-operator: u_ra (resource address); s (resource type)

Table 8.6 Instruction overview.

W = access width:	B = base address:
• 1 byte • 2 bytes • 4 bytes • 8 bytes • 16 bytes • reserved	• stack pointer (SP) • base pointer (BP) • global pointer (GP) • reserved

Table 8.7 Encoding of access width (W) and base address register selection (B).

8.4 Example 4: 16 bits, two functions

The principal instruction format layout of example 2 is applied to short instruction words of 16 bits in length. The resource address is limited to 6 bits. Such a resource address could support, for example, 16 resources with four parameters each or eight resources with eight parameters each. Applications: ReAl microcontrollers, small special processors, processing devices in FPGAs and the like. As the limitation to 16 bits leads to short displacement and immediates, double-lengt hvariants of p-, p_imm. and -operators have been provided. Fig. 8.4 provides an overview of the machine code, table 8.8 describes the instruction functions. For encoding of access width W and base address B refer to table 8.4.

Overview:

- Instruction length: 16 or 32 bits, respectively
- Resource address: 6 bits (flat address space)
- Resource type: 12 bits
- Immediate value: 6 or 20 bits, respectively
- Variable address (displacement): 4 or 18 bits, respectively. Up to 4 base address registers (B)
- Encoding of access width: hardwired in the resources (in case of 16 bit instructions) or in the instruction (in case of 32 bit instructions). Up to 4 variants (W)
- Special feature: the y-operator can activate two resources at the same time

operator	15 12	11 6	5 0	15 **2nd 16-bit-Wort** 0
p_l	0 0 1 0	resource	B displ.	W displacement
p_s	0 0 1 1	resource	displacement	–
p_imml	0 1 0 0	resource	immediate	W immediate (14)
p_imms	0 1 0 1	resource	immediate	–
a_l	0 1 1 0	resource	B displ.	W displacement
a_s	0 1 1 1	resource	displacement	–
l	1 0 0 0	2. resource	1. resource	
c	1 0 0 1	2. resource	1. resource	
d	1 0 1 0	2. resource	1. resource	
r	1 0 1 1	2. resource	1. resource	
y	1 1 0 0	2. resource	1. resource	
y_f	1 1 0 1	function	resource	
s	1 1 1 0	resource type (12)		
s_a	1 1 1 1	resource type (12)		resource (6)

Fig. 8.4 Instruction formats.

operator	**function**
p_l	fetch parameter out of system memory and move it into the selected resource (parameter passing). Content of variable address in memory = <base B + displacement> => resource. Long displacement (18 bits). Current access width encoded in W
p_s	fetch parameter out of system memory and move it into the selected resource (parameter passing). Content of variable address in memory = <base B + displacement> => resource. Short displacement (4 bits). Access width depending on the particular resource (hardwired)

operator	function
p_imml	load immediate value into the selected resource (parameter passing). Long immediate (20 bits). Current access width encoded in W. If access width W is greater than 16 bits, the immediate value will be sign-extended
p_imms	load immediate value into the selected resource (parameter passing). Short immediate (6 bits). Access width depending on the particular resource (hardwired). The immediate value will be sign-extended appropriately
a_l	store result from the selected resource into system memory (assign result). Resource => variable address in memory = <base B + displacement>. Long displacement (18 bits). Current access width encoded in W
a_s	store result from the selected resource into system memory (assign result). Resource => variable address in memory = <base B + displacement>. Short displacement (4 bits). Access width depending on the particular resource (hardwired)
l	move data between the selected resources (link). 1st resource => 2nd resource. Current access width encoded in W Zugriffsbreite gemäß Ressource oder fest)
c	establish a concatenation 1st resource => 2nd resource. The fields address an operand parameter and a result parameter
d	disconnect the concatenation 1st resource => 2nd resource; see also under c
r	return the selected resources to the resource pool. The fields address the function code parameters of the respective resources. Value = 0: no effect
y	initiate the information processing operations of the selected resources (yield). The fields address the function code parameters of the respective resources. Value = 0: no initiation
y_f	initiate the information processing operations in the selected resource according to the selected function code. Function code = 0: no initiation. Application: for resources with selectable functions
s	select a resource out of the resource pool according to the type field
s_a	select (request out of the resource pool) a resource according to the type field and assign the selected resource address. This resource address is the base of resource enumeration in subsequent s-operators (+1 increment). Type = 0: no resource selected (but a resource address other than zero will be assigned)

Table 8.8 Instruction overview.

8.5 Example 5: 16 bits, longer addresses

The principal instruction format layout of example 3 is modified to short instruction words of 16 bits in length, containing address fields that are appropriately shorter, but still useful. A resource address of 12 bits could support, for example, up to 512 resources with eight parameters each. As each instruction can contain only one address or data field, four buffer registers are provided, similar to example 3, that can be loaded with u-operators. Some operators require therefore two instructions. Applications: software emulation (virtual machines) in the performance range of typical embedded systems, ReAl microprocessors and the like. Fig. 8.5 provides an overview of the machine code. For more details refer to the tables 8.5 to 8.7 (example 3).

Overview:

- Instruction length: 16 bits
- Resource address: 12 bits (flat address space)
- Resource type: 10 bits
- Immediate value: 11 bits
- Variable address (displacement): 9 bits; up to 4 base address registers (B)
- Encoding of access width: in the instruction; up to 4 variants (W)
- Buffering: four buffer registers

operator	15		13		11	9	8	0
others (reserved)	0	0	0	0	*)		res.	
s					11		resource type (10)	
r	0	0	0	1	resource (12)			
p_imm2	0	0	1	0	resource (12)			
y	0	0	1	1	resource (12)			
l_2	0	1	0	0	resource (12)			
c_2	0	1	0	1	resource (12)			
d_2	0	1	1	0	resource (12)			
u_rs	0	1	1	1	resource (12)			
a_2	1	0	0	W	B	displacement (9)		
u_va	1	0	1	W	B	displacement (9)		
u_imm	1	1	0	W	immediate (11)			
p_2	1	1	1	0	resource (12)			
u_ra	1	1	1	1	resource address (12)			

*): Codes 00, 01, 10

Fig. 8.5 Instruction formats. Instruction formats.

8.6 Example 6: 16 bits, some functions can be controlled in parallel

This example demonstrates that even short instruction words (16 bits) can control some functions in parallel. The instruction set supports up to 64 resources with up to eight parameters each (split address space). As the content of some operators has to be supplied in pieces, three buffer registers have to be provided in the platform. These registers can be loaded by additional operators u_rs1, u_rs2 and u_ra. The y-operator can activate up to 8 resources at the same time (by means of a bit mask). Applications: ReAl microcontrollers and microprocessors, special processors, processing devices in FPGAs and the like. Table 8.9 shows the content of the buffer registers. Fig. 8.6 illustrates the basic instruction format. Table 8.10 describes the instruction functions. For encoding of base addres register selection (B) refer to table 8.4.

Overview:

- Instruction length: 16 bits
- Resource address: 6 bits (split address space)
- Parameter address: 3 bits
- Resource type: 10 bits
- Immediate value: 9 bits
- Variable address (displacement): 9 bits; up to 4 base address registers (B)
- Encoding of access width: in the resources (hardwired)
- Buffering: three buffer registers
- Simultaneous initiation of:
  - two parameter transports between the resources
  - two concatenation control functions
  - activation of up to eight resources
  - entry of two resource addresses into a buffer register

	11 6	5 0
buffer register 1	1st source resource	1st destination resource
buffer register 2	2nd source resource	2nd destination resource
buffer register 3	–	resource address

Table 8.9 Buffer registers.

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.
others (reserved)	0	0	0	0	1	0	0	res.						
s	0	0	0	0	1	0	1	resource type (9)						
u_ra	0	0	0	0	1	1	resource address (6)							
l	0	0	0	1	1st SP			1st DP			2nd SP		2nd DP	
c	0	0	1	0	1st SP			1st DP			2nd SP		2nd DP	
d	0	0	1	1	1st SP			1st DP			2nd SP		2nd DP	
r	0	1	0	0	1st resource						2nd resource			
u_rs1	0	1	0	1	1st source resource (6)						1st dest. ressource (6)			
u_rs2	0	1	1	0	2nd source resource (6)						2nd dest. ressource (6)			
p_imm	0	1	1	1	immediate (9)								parameter	

SP = source parameter; dp = destination parameter

Fig. 8.6 Instruction formats.

operator	function
p	fetch parameter out of system memory and move it into the selected resource (parameter passing). Content of variable address in memory = <base B + displacement> => parameter in resource. The resource address is expected in buffer register 1 (1st destination resource)
a	store result from the selected resource into system memory (assign result). Parameter in resource => variable address in memory = <base B + displacement>. The resource address is expected in buffer register 1 (1st source resource)
y	initiate the information processing operations of the selected resources (yield). The field *ress. sel* selects one group of eight out of the 64 resources supported. One of the eight control bits (8...1) is assigned to each resource of the selected group. Control bit = 1: initiate operation, control bit = 0: don't initiate operation
s	select a resource out of the resource pool according to the type field. The resource address is expected in buffer register 3. After resource selection, the content of buffer register 3 will be incremented by one. Operator s_a can be emulated by an operator sequence u_ra (resource address); s (resource type)
u_ra	load the content of the address field into buffer register 3 (resource address parameter for the s-operator). This resource address is the base of resource enumeration in subsequent s-operators (+1 increment)
l	move data between the selected resources (link). Up to two transport operations will be executed: 1st source parameter => 1st destination parameter; 2nd source parameter => 2nd destination parameter. The resource addresses of the 1st transport are expected in buffer register 1, the resource address of the 2nd transport in buffer register 2. Source parameter = 0: no transport
c	establish up to two concatenations.1st source parameter => 1st destination parameter; 2nd source parameter => 2nd destination parameter. The resource addresses for the 1st concatenation are expected in buffer register 1, the resource address for the 2nd concatenation in buffer register 2. Source parameter = 0: no concatenation
d	disconnect up to two concatenations. 1st source parameter => 1st destination parameter; 2nd source parameter => 2nd destination parameter. The resource addresses of the 1st concatenation are expected in buffer register 1, the resource address of the 2nd concatenation in buffer register 2. Source parameter = 0: no effect
r	return up to two resources to the resource pool. Resource address = 0: no effect
u_rs1	load two resource addresses into buffer register 1
u_rs2	load two resource addresses into buffer register 2
p_imm	load immediate value into the selected resource (parameter passing). If the access width of the selected parameter is greater than 9 bits, the immediate value will be sign-extended. The resource address is expected in the 1st buffer register (1st destination resource)

Table 8.10 Instruction overview.

8.7 Example 7: 64 bits, many functions can be controlled in parallel

The last example demonstrates an instruction format that enables encoding of many parallel executable functions. For this purpose, longer instruction words are required (64 bits). Similar to example 6, this instruction set supports up to 64 resources with up to eight parameters each (split address space). The platform contains three buffer registers. These registers can be loaded by additional operators u_rs1, u_rs2 and u_ra. The y-operator can activate up to 60 resources at the same time (by means of a bit mask). Applications: special processors, processing devices in FPGAs and so on. Table 8.11 shows the content of the buffer registers. Fig. 8.7 illustrates the basic instruction word format. Fig. 8.8 contains details of the parameter fields in the operators p, p_imm and a. Table 8.12 gives an overview over the various operation codes. Fig. 8.9 illustrates the basic instruction format. Table 8.13 illustrates details of the bit positions 59...0. Table 8.14 describes the instruction functions.

Overview:

- Instruction length: 64 bits
- Resource address: 6 bits (split address space)
- Parameter address: 3 bits
- Resource type: 10 bits
- Immediate value: 12 bits
- Variable address (displacement): 10 bits; up to 4 base address registers (B)
- Encoding of access width: in the resources (hardwired)
- Buffering: three buffer registers
- Simultaneous initiation of:
  - 10 parameter transports between the resources
  - 10 concatenation control functions
  - activation of up to 60 resources
  - entry of 10 resource addresses into a buffer register
  - allocation of up to 6 resources
  - returning of up to 10 resources to the resource pool

buffer register 1	s1	d1	s2	d2	s3	d3	s4	d4	s5	d5
buffer register 2	s6	d6	s7	d7	s8	d8	s9	d9	s10	d10
buffer register 3	ra 1		ra 2		ra 3		ra 4		ra 5	ra 6

Table 8.11 Buffer registers.

63 60	59 0
opcode	addresses, immediate values or control information

Fig. 8.7 Principal instruction word format.

operation	14 12	11 3	2 0
p, a	B	displacement (10)	parameter
p_imm	immediate (12)		parameter

Fig. 8.8 Parameter fields in p-, p_imm- and a-operators. For encoding of base address register selection (B) refer to table 8.4.

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	others *)	7	d	B	p_imm	F	u_rs1

*): reserved.

Table 8.12 Operation codes.

operator	**bits 59...0** (see Table 8.13 for details)									
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_rs1	s1	d1	s2	d2	s3	d3	s4	d4	s5	d5
u_rs2	s6	d6	s7	d7	s8	d8	s9	d9	s10	d10
y	60 control bits									
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)

Fig. 8.9 Instruction formats.

operator	the bits 59...0 contain:
p	4 parameter fields (parm1...4) of 15 bits each. Refer to fig. 8.8
a	4 parameter fields (parm1...4) of 15 bits each. Refer to fig. 8.8
y_1	60 control bits for activation of the resources 59...0
y_2	60 control bits for activation of the resources 63...60 and 55...0
s	6 resource type fields (rt1...10) of 10 bits each
u_ra	6 resource addresses of 10 bits each (only 6 bits are utilized, respectively)

operator	the bits 59...0 contain:
l, c, d	10 parameter address pairs (s/d1...s/d10) of 6 bits each. s = source parameter (3 bits), d = destination parameter (3 bits)
r	10 resource addresses (r1...r10) of 6 bits each
u_rs1	5 resource address pairs (s1, d1...s5, d5) of 12 bits each (for buffer register 1)
u_rs2	5 resource address pairs (s6, d6...s10, d10) of 12 bits each (for buffer register 2)
p_imm	4 parameter fields (parm1...4) of 15 bits each. Refer to fig. 8.8

Table 8.13 The bits 59...0 in the various instruction formats.

operator	function
p	fetch parameter out of system memory and move it into the selected resources (parameter passing). 4 operand transports content of variable address in memory = <base B + displacement> => parameter in resource. The resource addresses are expected in buffer register 1 (destination resources d1...d4). Parameter = 7: no transport
a	store result from the selected resources into system memory (assign result). 4 result transports parameter in resource => variable address in memory = <base B + displacement>. The resource addressses are expected in buffer register 1 (source resources s1...s4). Parameter = 7: no transport
y_1	initiate the information processing operations of the first 60 resources (59...0). One of the 60 control bits (59...0) is assigned to each of those resources. Control bit = 1: initiate operation, control bit = 0: don't initiate operation
y_2	initiate the information processing operations of the last 4 and the first 56 resources (63...60, 55...0). One of the 60 control bits (59...0) is assigned to each of those resources. Control bit = 1: initiate operation, control bit = 0: don't initiate operation
s	select 6 resources out of the resource pool according to the type fields (rt1...6). Resource type = 0: no selection. The resource addresses are taken from buffer register 3. There is no automatic address increment. Resources are to be selected always by sequences u_ra; s (like the s_a operators of other examples)
u_ra	load the content of the address fields into buffer register 3 (resource address parameters for the s-operator)
l	move data between the selected resources (link). Up to 10 transport operations will be executed: source parameter => destination parameter (s1 => d1...s10 => d10). The resource addresses for the first 5 transport operations (1...5) are expected in buffer register 1, for the second 5 transport operations (6...10) in buffer register 2. Source parameter = 0: no transport
c	establish up to 10 concatenations source parameter => destination parameter (s1 => d1...s10 => d10). The resource addresses for the first 5 concatenations (1...5) are expected in buffer register 1, for the second 5 concatenations (6...10) in buffer register 2. Source parameter = 0: no concatenation

operator	function
d	disconnect up to 10 concatenations source parameter => destination parameter (s1 => d1...s10 => d10). The resource addresses of the first 5 concatenations (1...5) are expected in buffer register 1, of the second 5 concatenations (6...10) in buffer register 2. Source parameter = 0: no effect
r	return up to 10 resources (r1...r10) to the resource pool. Resource address = 0: no effect
u_rs1	load 5 ressource address pairs into buffer register 1 (s1, d1...s5, d5)
u_rs2	load 10 ressource address pairs into buffer register 2 (s6, d6...s10, d10)
p_imm	load 4 immediate values into the selected resources (parameter passing). If the access width of the selected parameter is greater than 12 bits, the immediate value will be sign-extended. The resource addresses are expected in the 1st buffer register (destination resources d1...d4). Parameter = 7: no transport

Table 8.14 Instruction overview.