DLX procesors un instrukcijas

kurss *Digitālo iekārtu projektēšana*
lasa
Leo Seļāvo

DLX arhitektūra

- R2000 – 32-bitu RISC procesors, MIPS arhitektūra
- DLX – radinieks, pārsvarā domāts studijām
  - 32 bitu dati
  - 32 bitu instrukcijas
  - *Load* – *Store* arhitektūra
  - 5 līmeņu Konvejiera princips
    - IF – Instruction Fetch
    - ID – Instruction Decode
    - EX – Execute
    - MEM – Memory access
    - WB – Write Back

Atmiņa

DLX ir “Big Endian”

- Little Endian
  - *(DEC, Intel)*

Addr 0	3	2	1	0
Addr 4	7	6	5	4

- Big Endian
  - *(IBM, Motorola, ...)*

Addr 0	0	1	2	3
Addr 4	4	5	6	7

Reģistri

- 32 reģistri R0 - R31, 32 biti katrs
- R0 == 0, vienmēr
- R31 – īpašs, atkarībā no instrukcijas

- PC – Program Counter reģistrs (32 biti)
  - RESET PC = 0

Instrukciju tipi

Formāts	Biti					
	31 - 26	25 - 21	20 - 16	15 - 11	10 - 6	5 - 0
R-type	0x0	Rs1	Rs2	Rd	*unused*	*opcode*
I-type	*opcode*	Rs1	Rd	*immediate*		
J-type	*opcode*	*value*				

Rs – Source Register (avota reg.)
Rd – Destination Register (mērķa reg.)
opcode – operācijas kods
immediate – skaitliska vērtība
value – pārejas adreses vērtība

Instrukcijas: aritmētika un loģika

Instrukcija	Apraksts	Formāts	Opcode	Operācija (C programmēšanas stilā)
ADD	add	R	0x20	Rd = Rs1 + Rs2
ADDI	add immediate	I	0x08	Rd = Rs1 + extend(immediate)
SUB	subtract	R	0x22	Rd = Rs1 - Rs2
SUBI	subtract immediate	I	0x0a	Rd = Rs1 - extend(immediate)
AND	and	R	0x24	Rd = Rs1 & Rs2
ANDI	and immediate	I	0x0c	Rd = Rs1 & immediate
OR	or	R	0x25	Rd = Rs1 | Rs2
ORI	or immediate	I	0x0d	Rd = Rs1 | immediate
XOR	exclusive or	R	0x26	Rd = Rs1 ^ Rs2
XORI	exclusive or immediate	I	0x0e	Rd = Rs1 ^ immediate

Instrukcijas: nobīde

Instrukcija	Apraksts	Formāts	Opcode	Operācija (C programmēšanas stilā)
SLL	shift left logical	R	0x04	Rd = Rs1 << (Rs2 % 8)
SLLI	shift left logical immediate	I	0x14	Rd = Rs1 << (immediate % 8)
SRA	shift right arithmetic	R	0x07	as SRL & see below
SRAI	shift right arithmetic immediate	I	0x17	as SRLI & see below
SRL	shift right logical	R	0x06	Rd = Rs1 >> (Rs2 % 8)
SRLI	shift right logical immediate	I	0x16	Rd = Rs1 >> (immediate % 8)

Instrukcijas: pārbaude

Instrukcija	Apraksts	Formāts	Opcode	Operācija (C programmēšanas stilā)
SEQ	set if equal	R	0x28	Rd = (Rs1 == Rs2 ? 1 : 0)
SEQI	set if equal to immediate	I	0x18	Rd = (Rs1 == extend(immediate) ? 1 : 0)
SLE	set if less than or equal	R	0x2c	Rd = (Rs1 <= Rs2 ? 1 : 0)
SLEI	set if less than or equal to immediate	I	0x1c	Rd = (Rs1 <= extend(immediate) ? 1 : 0)
SLT	set if less than	R	0x2a	Rd = (Rs1 < Rs2 ? 1 : 0)
SLTI	set if less than immediate	I	0x1a	Rd = (Rs1 < extend(immediate) ? 1 : 0)
SNE	set if not equal	R	0x29	Rd = (Rs1 != Rs2 ? 1 : 0)
SNEI	set if not equal to immediate	I	0x19	Rd = (Rs1 != extend(immediate) ? 1 : 0)

Instrukcijas: pāreja

Instrukcija	Apraksts	Formāts	Opcode	Operācija (C programmēšanas stilā)
BEQZ	branch if equal to zero	I	0x04	PC += (Rs1 == 0 ? extend(immediate) : 0)
BNEZ	branch if not equal to zero	I	0x05	PC += (Rs1 != 0 ? extend(immediate) : 0)
J	jump	J	0x02	PC += extend(value)
JAL	jump and link	J	0x03	R31 = PC + 4 ; PC += extend(value)
JALR	jump and link register	I	0x13	R31 = PC + 4 ; PC = Rs1
JR	jump register	I	0x12	PC = Rs1

- JR un JALR neizmanto Rd un *immediate*

Instrukcijas: Load - Store

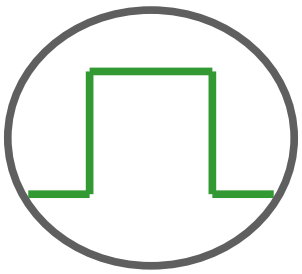
Instrukcija	Apraksts	Formāts	Opcode	Operācija (C programmēšanas stilā)
LW	load word	I	0x23	Rd = MEM[Rs1 + extend(immediate)]
SW	store word	I	0x2b	MEM[Rs1 + extend(immediate)] = **Rd**
LHI	load high bits	I	0x0f	Rd = immediate << 16

- SW lasa no Rd !
- LHI ielasa tikai augšējos 16 bitus no konstantes.
  Lai ielasītu 32 bitus no konstantes (piemēram, 0x12345678), jāizpilda:

```
LHI   R1, #0x1234
ORI   R1, R1, #0x5678
```

Instrukcijas: vēl...

- Kā vēl trūkst?
  - MULT, DIV
  - Floating Point
  - NOP
  - HALT
  - Traps, Exceptions



Vairāk informācija par DLX

- http://www.csee.umbc.edu/courses/undergraduate/411/spring96/dlx.html