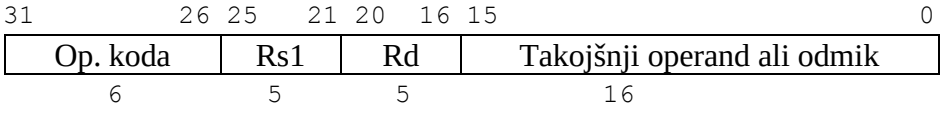
Zbirnik podpira naslednje psevdo-ukaze

```
.data                  - začetek podatkovnega segmenta
.text                  - začetek ukaznega segmenta
.code                  - začetek ukaznega segmenta (enako kot .text)
.org    <n>            - določen začetni naslov
.space  <n>            - rezerviraj n lokacij (pusti naključno vrednost)
.asciiz <s>            - ASCII niz zaključen z ničlo
.ascii  <s>            - ASCII niz
.align  <n>            - poravnaj naslov da bo deljiv z n
.word   <n1>,<n2>..    - določi eno ali več zaporednih 32-bitnih števil
.byte   <n1>,<n2>..    - določi eno ali več zaporednih 8-bitnih števil
.word64 <n1>,<n2>..    - določi eno ali več zaporednih 64-bitnih števil
.word16 <n1>,<n2>..    - določi eno ali več zaporednih 16-bitnih števil
.double <n1>,<n2>..    - določi eno ali več zaporednih 64-bitnih števil
                         v plavajoči vejici (IEEE754)
```

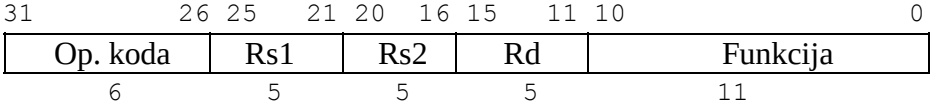
<n> predstavlja število kot na primer 24
<s> predstavlja niz kot na primer "fred"
<n1>,<n2>.. predstavlja števila ločena z vejicami.

Formata ukazov procesorja HIP

Format 1:



Format 2:



Nabor ukazov procesorja HIP

Mnemo-nik	Ime ukaza	Splošna oblika	Opis	Primer ukaza	Strojna koda primera	#up
add	add (signed)	add Rd,Rs1,Rs2	Rd <- Rs1 + Rs2	add r1,r2,r3	C0430800	4
addi	add immediate	addi Rd,Rs1,#n	Rd <- Rs1 + n, sign ext.	addi r1,r2,#0x1234	00411234	4
addu	add unsigned	addu Rd,Rs1,Rs2	Rd <- Rs1 + Rs2,unsigned	addu r1,r2,r3	C8430800	4
addui	add unsigned immediate	addui Rd,Rs1,#n	Rd <- Rs1 + n, unsigned	addui r1,r2,#0x1234	08411234	4
and	logical and	and Rd,Rs1,Rs2	Rd <- Rs1 and Rs2	and r1,r2,r3	D0430800	4
andi	and immediate	andi Rd,Rs1,#n	Rd <- Rs1 and n	andi r1,r2,#0x1234	10411234	4
beq	branch if equal	beq Rd,LABEL	Rd == 0 => PC <- LABEL (relative)	beq r0,L1(L1 na za 4 večjem naslovu)	9C000000	3
bne	branch if not equal	bne Rd,LABEL	Rd != 0 =>PC <- LABEL (relative)	bne r0,L1(L1 na za 4 večjem naslovu)	8C000000	3
call	call subroutine	call Rd,LABEL(Rs1)	Rd <- PC + 4,PC <- LABEL + Rs1	call r1,SR(r0) (SR na naslovu 0xD4)	B40100D4	5
di	disable interrupts	di	I <- 0	di	D0000001	3
ei	enable interrupts	ei	I <- 1	ei	CC000001	3
halt	halts execution	halt		halt	7C000000	3
j	jump	j LABEL(Rs1)	PC <- LABEL + Rs1	j L1(r0)(L1 na naslovu 0x30)	B0000030	3
lb	load byte	lb Rd,OFFSET(Rs1)	Rd <- MEM[OFFSET+Rs1],sign extended	lb r1,0(r2)	90410000	6
lbu	load byte unsigned	lbu Rd,OFFSET(Rs1)	Rd <- MEM[OFFSET+Rs1],unsigned	lbu r1,0(r2)	80410000	6
lh	load halfword (16 bit)	lh Rd,OFFSET(Rs1)	Rd <- MEM[OFFSET+Rs1],sign extended	lh r1,0(r2)	94410000	6

lhi	load high immediate	lhi Rd,#n	Rd(31..16) <- n,Rd(15..0) <- 0	lhi r1,#0x12340000	1C011234	4
lhu	load halfword unsigned (16 bit)	lhu Rd,OFFSET(Rs1)	Rd <- MEM[OFFSET+Rs1],unsigned	lhu r1,0(r2)	84410000	6
lw	load word (32 bit)	lw rd,OFFSET(Rs1)	Rd <- MEM[OFFSET+Rs1]	lw r1,0(r2)	98410000	6
mover	move from EPC to register	mover Rd	Rd <- EPC	mover r1	D4000801	4
movre	move from register to EPC	movre Rs1	EPC <- Rs1	movre r0	D8000001	3
nop	no operation	nop		nop	00000000	4
not	not(1's complement)	not Rd,Rs1,Rs2	Rd <- not Rs1	not r1,r2,r3	F8430800	4
or	logical or	or Rd,Rs1,Rs2	Rd <- Rs1 or Rs2	or r1,r2,r3	D4430800	4
ori	logical or immediate	ori Rd,Rs1,#n	Rd <- Rs1 or n	ori r1,r2,#0x1234	14411234	4
rfe	return from exception	rfe	PC <- EPC	rfe	BC000000	3
sb	store byte	sb OFFSET(Rs1),Rd	MEM[OFFSET+Rs1] <- Rd	sb 0(r2),r3	A0430000	4
seq	set if equal	seq Rd,Rs1,Rs2	Rd <- Rs1 == Rs2	seq r1,r2,r3	E0430800	4
seqi	set if equal immediate	seqi Rd,Rs1,#n	Rd <- Rs1 == n	seqi r1,r2,#0x1234	20411234	4
sgt	set if greater than	sgt Rd,Rs1,Rs2	Rd <- Rs1 > Rs2	sgt r1,r2,r3	EC430800	4
sgti	set if greater than immediate	sgti Rd,Rs1,#n	Rd <- Rs1 > n	sgti r1,r2,#0x1234	2C411234	4
sgtu	set if greater than unsigned	sgtu Rd,Rs1,Rs2	Rd <- Rs1 > Rs2unsigned	sgtu r1,r2,r3	F4430800	4
sgtui	set if greater than unsigned immediate	sgtui Rd,Rs1,#n	Rd <- Rs1 > n unsigned	sgtui r1,r2,#0x1234	34411234	4
sh	store halfword (16 bit)	sh OFFSET(Rs1),Rd	MEM[OFFSET+Rs1] <- Rd	sh 0(r2),r3	A4430000	4
sll	shift left logical	sll Rd,Rs1,Rs2	Rd <- Rs1 << Rs2, b0 <- 0	sll r1,r2,r3	C0430801	4
slli	shift left logical immediate	slli Rd,Rs1,#n	Rd <- Rs1 << n, b0 <- 0	slli r1,r2,#0x1234	40411234	4
slt	set if less than	slt Rd,Rs1,Rs2	Rd <- Rs1 < Rs2	slt r1,r2,r3	E8430800	4
slti	set if less than immediate	slti Rd,Rs1,#n	Rd <- Rs1 < n	slti r1,r2,#0x1234	28411234	4
sltu	set if less than unsigned	sltu Rd,Rs1,Rs2	Rd <- Rs1 < Rs2unsigned	sltu r1,r2,r3	F0430800	4
sltui	set if less than unsigned immediate	sltui Rd,Rs1,#n	Rd <- Rs1 < n unsigned	sltui r1,r2,#0x1234	30411234	4
sne	set if not equal	sne Rd,Rs1,Rs2	Rd <- Rs1 != Rs2	sne r1,r2,r3	E4430800	4
snei	set if not equal immediate	snei Rd,Rs1,#n	Rd <- Rs1 != n	snei r1,r2,#0x1234	24411234	4
sra	shift right arithmetic	sra Rd,Rs1,Rs2	Rd <- Rs1 >> Rs2, b31 <- b31	sra r1,r2,r3	C8430801	4
srai	shift right arithmetic immediate	srai Rd,Rs1,#n	Rd <- Rs1 >> n, b31 <- b31	srai r1,r2,#0x1234	48411234	4
srl	shift right logical	srl Rd,Rs1,Rs2	Rd <- Rs1 >> Rs2, b31 <- 0	srl r1,r2,r3	C4430801	4
srli	shift right logical immediate	srli Rd,Rs1,#n	Rd <- Rs1 >> n, b31 <- 0	srli r1,r2,#0x1234	44411234	4
sub	subtract	sub Rd,Rs1,Rs2	Rd <- Rs1 - Rs2	sub r1,r2,r3	C4430800	4
subi	subtract immediate	subi Rd,Rs1,#n	Rd <- Rs1 - n, sign extended	subi r1,r2,#0x1234	04411234	4
subu	subtract unsigned	subu Rd,Rs1,Rs2	Rd <- Rs1 - Rs2	subu r1,r2,r3	CC430800	4
subui	subtract unsigned immediate	subui Rd,Rs1,#n	Rd <- Rs1 - n, unsigned	subui r1,r2,#0x1234	0C411234	4
sw	store word (32 bit)	sw OFFSET(Rs1),Rd	MEM[OFFSET+Rs1] <- Rd	sw 0(r2),r3	A8430000	4
trap	software interrupt	trap #n	EPC <- PC,PC <- M[FFFFFF00+4*n]	trap #2	B8000002	6
xor	exclusive or	xor Rd,Rs1,Rs2	Rd <- Rs1 xor Rs2	xor r1,r2,r3	D8430800	4
xori	exclusive or immediate	xori Rd,Rs1,#n	Rd <- Rs1 xor n	xori r1,r2,#0x1234	18411234	4