Understanding Program Counter

Sparsh Mittal

An example of add instruction (R-format)

funct7	rs2	rs1	funct3	rd	opcode
7 bits	5 bits	5 bits	3 bits	5 bits	7 bits

```
add x9,x20,x21
```

0	21	20	0	9	51

0000000	10101	10100	000	01001	0110011

$0000\ 0001\ 0101\ 1010\ 0000\ 0100\ 1011\ 0011_{two}$

This slide shows how in RISC-V, each instruction takes 32 bits (4 bytes).

Program counter (PC) and how it changes through program execution.

- Example (RV32):

Memory Address:	Instruction Stored:
0x00000000:	addi x10 x0 6
0x00000004:	addi x11 x0 10
0x00000008:	add x13 x10 x11
0x0000000c:	sub x14 x11 x10

Program counter:
0x00000000

Program counter (PC) and how it changes through program execution.

- Example (RV32):

Memory Address:	Instruction Stored:
0x00000000:	addi x10 x0 6
0x00000004:	addi x11 x0 10
0x00000008:	add x13 x10 x11
0x0000000c:	sub x14 x11 x10

Program counter:
0x00000004

Program counter (PC) and how it changes through program execution.

- Example (RV32):

Memory Address:	Instruction Stored:
0x00000000:	addi x10 x0 6
0x00000004:	addi x11 x0 10
0x00000008:	add x13 x10 x11
0x0000000c:	sub x14 x11 x10

Program counter:
0x00000008

Program counter (PC) and how it changes through program execution.

- Example (RV32):

Memory Address:	Instruction Stored:
0x00000000:	addi x10 x0 6
0x00000004:	addi x11 x0 10
0x00000008:	add x13 x10 x11
0x0000000c:	sub x14 x11 x10

Program counter:
0x0000000c

Program counter (PC) and how it changes through program execution.

- Example containing a function call (RV32):

Label:	Memory Address:	Instruction Stored:
_start:	0x00000000:	li a0 0x00000006
	0x00000004:	li a1 0x0000000a
	0x00000008:	jal add_func
	0x0000000c:	j exit
add_func:	0x00000010:	add a3 a0 a1
	0x00000014:	ret
exit:	0x00000018:	nop

- At the beginning, PC will be initialized to the first location of the program that is 0x0000000 as in the previous example.
- The increments to the PC will follow the sequence, unless we encounter a control-flow changing instruction, such as ‘Unconditional Jump’ and ‘Conditional Jump’ instructions.
- These instructions will change the PC contents to different locations.

Program counter (PC) and how it changes through program execution.

- Example containing a function call (RV32):

Label:	Memory Address:	Instruction Stored:
_start:	0x00000000:	li a0 0x00000006
	0x00000004:	li a1 0x0000000a
	0x00000008:	jal add_func
	0x0000000c:	j exit
add_func:	0x00000010:	add a3 a0 a1
	0x00000014:	ret
exit:	0x00000018:	nop

Program Counter:
0x00000000
Return Address:
0x00000000

Program counter (PC) and how it changes through program execution.

- Example containing a function call (RV32):

Label:	Memory Address:	Instruction Stored:
_start:	0x00000000:	li a0 0x00000006
	0x00000004:	li a1 0x0000000a
	0x00000008:	jal add_func
	0x0000000c:	j exit
add_func:	0x00000010:	add a3 a0 a1
	0x00000014:	ret
exit:	0x00000018:	nop

Program Counter:
0x00000004
Return Address:
0x00000000

Program counter (PC) and how it changes through program execution.

- Example containing a function call (RV32):

Label:	Memory Address:	Instruction Stored:
_start:	0x00000000:	li a0 0x00000006
	0x00000004:	li a1 0x0000000a
	0x00000008:	jal add_func
	0x0000000c:	j exit
add_func:	0x00000010:	add a3 a0 a1
	0x00000014:	ret
exit:	0x00000018:	nop

Program Counter:
0x00000008
Return Address:
0x00000000

Program counter (PC) and how it changes through program execution.

- Example containing a function call (RV32):

Label:	Memory Address:	Instruction Stored:
_start:	0x00000000:	li a0 0x00000006
	0x00000004:	li a1 0x0000000a
	0x00000008:	jal add_func
	0x0000000c:	j exit
add_func:	0x00000010:	add a3 a0 a1
	0x00000014:	ret
exit:	0x00000018:	nop

Program Counter:
0x00000010
Return Address:
0x0000000c

Program counter (PC) and how it changes through program execution.

- Example containing a function call (RV32):

Label:	Memory Address:	Instruction Stored:
_start:	0x00000000:	li a0 0x00000006
	0x00000004:	li a1 0x0000000a
	0x00000008:	jal add_func
	0x0000000c:	j exit
add_func:	0x00000010:	add a3 a0 a1
	0x00000014:	ret
exit:	0x00000018:	nop

Program Counter:
0x00000014
Return Address:
0x0000000c

Program counter (PC) and how it changes through program execution.

- Example containing a function call (RV32):

Label:	Memory Address:	Instruction Stored:
_start:	0x00000000:	li a0 0x00000006
	0x00000004:	li a1 0x0000000a
	0x00000008:	jal add_func
	0x0000000c:	j exit
add_func:	0x00000010:	add a3 a0 a1
	0x00000014:	ret
exit:	0x00000018:	nop

Program Counter:
0x0000000c
Return Address:
0x0000000c

Program counter (PC) and how it changes through program execution.

- Example containing a function call (RV32):

Label:	Memory Address:	Instruction Stored:
_start:	0x00000000:	li a0 0x00000006
	0x00000004:	li a1 0x0000000a
	0x00000008:	jal add_func
	0x0000000c:	j exit
add_func:	0x00000010:	add a3 a0 a1
	0x00000014:	ret
exit:	0x00000018:	nop

Program Counter:
0x00000018
Return Address:
0x0000000c

Example instruction	Instruction name	Meaning
`jal x1,offset`	Jump and link	`Regs[x1]←PC+4; PC←PC+(offset<<1)`
`jalr x1,x2,offset`	Jump and link register	`Regs[x1]←PC+4; PC←Regs[x2]+offset`
`beq x3,x4,offset`	Branch equal zero	`if (Regs[x3]==Regs[x4]) PC←PC+(offset<<1)`
`bgt x3,x4,name`	Branch not equal zero	`if (Regs[x3]>Regs[x4]) PC←PC+(offset<<1)`

Figure A.27 Typical control flow instructions in RISC-V. All control instructions, except jumps to an address in a register, are PC-relative.