

CSCI 370: Computer Architecture

Y86-64 Reference

Instruction Format

halt	0 0	
nop	1 0	
rrmovq rA , rB	2 0 rA rB	
cmovXX rA , rB	2 fn rA rB	
irmovq V , rB	3 0 F rB	V
rmmovq rA , D(rB)	4 0 rA rB	D
mrmovq D(rB) , rA	5 0 rA rB	D
addq rA , rB	6 0 rA rB	
subq rA , rB	6 1 rA rB	
andq rA , rB	6 2 rA rB	
xorq rA , rB	6 3 rA rB	
jmp Dest	7 0	Dest
jXX Dest	7 fn	Dest
call Dest	8 0	Dest
ret	9 0	
pushq rA	A 0 rA	F
popq rA	B 0 rA	F

fn Codes

1	le	3	e	5	ge
2	l	4	ne	6	g

Registers

ID	Enc	Usage	
%rdi	7	arg1	caller-saved
%rsi	6	arg2	
%rdx	2	arg3	
%rcx	1	arg4	
%r8	8	arg5	
%r9	9	arg6	callee-saved
%rax	0	return	
%r10	A	general	
%r11	B	general	
%rbx	3	general	
%r12	C	general	
%r13	D	general	
%r14	E	general	
%rsp	4	stack ptr	
%rbp	5	base ptr	
	F	no reg	

Status Conditions

AOK	1	Normal
HLT	2	Halt Encountered
ADR	3	Bad Address
INS	4	Invalid Instruction

HCL Y86-64 Hardware Registers

stage	register(s)	description
Fetch	icode,ifun	Read instruction byte
	rA,rB	Read register byte
	valC	Read constant word
	valP	Compute next PC
Decode	valA,srcA	Read operand A
	valB,srcB	Read operand B
Execute	valE	Perform ALU operation
	cnd	Set/Use Condition Code
Memory	valM	Memory Read/Write
Writeback	dstE	Write back ALU result
	dstM	Write back Mem result
PC Update	PC	Update PC

Y86-64 Data Example

```
.align 8
Array:
.quad 0x0000000000000001
.quad 0x0000000000000002
.quad 0x0000000000000003
.quad 0x0000000000000004
```

Assembly Translation Example

```
/* find number of elements in null-terminated list */
long len(long* a) {
    long len;
    for (len = 0; a[len]; ++len)
        ;
    return len;
}

len:
    irmovq $1, %r8           # Constant 1
    irmovq $8, %r9           # Constant 8
    irmovq $0, %rax          # len = 0
    mrmovq (%rdi), %rdx      # val = *a
    andq %rdx, %rdx          # Test val
    je Done                  # If zero, goto Done

Loop:
    addq %r8, %rax           # len++
    addq %r9, %rdi           # a++
    mrmovq (%rdi), %rdx      # val = *a
    andq %rdx, %rdx          # Test val
    jne Loop                 # If !0, goto Loop

Done:
    ret
```