CS & IT ENGINEERING

COMPUTER ORGANIZATION AND ARCHITECTURE

Instruction & Addressing Modes

Lecture No.- 04

By- Vishvadeep Gothi sir

Recap of Previous Lecture

Topic Multiple Instruction Support

Topic Variable Size Instructions

Topics to be Covered

- **Topic** Instruction Construction for CPU
- **Topic** Instruction Cycle
- **Topic** Fetch & Execution Cycle
- **Topic** Branch Instruction

[NAT] GATE-2018

Int Reg. field ⇒ 4 bits

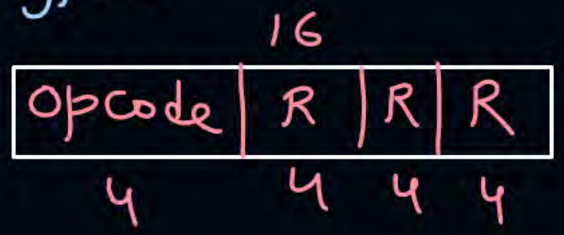
float Reg. field = 6 bits

#Q. A processor has 16 integer registers (R0, R1, … , R15) and 64 floating point registers (F0, F1, … , F63). It uses a 2-byte instruction format. There are four categories of instructions: Type-1, Type-2, Type-3, and Type 4. Type-1 category consists of four instructions, each with 3 integer register operands (3Rs). Type-2 category consists of eight instructions, each with 2 floating point register operands (2Fs). Type-3 category consists of fourteen instructions, each with one integer register operand and one floating point register operand (1R+1F). Type-4 category consists of N instructions, each with a floating-point register operand (1F).

The maximum value of N is 32 ?

Inst^n size = 2 bytes = 16 bits

Type 1

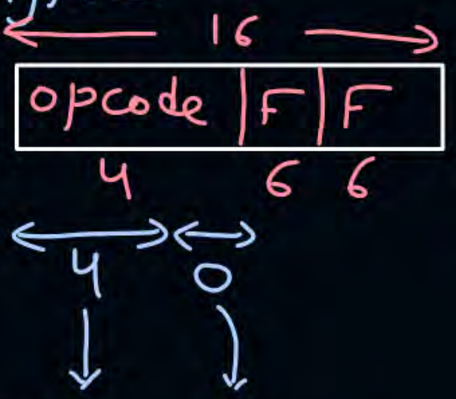


max = $2^4 = 16$

used = 4

unused = 12

Type 2

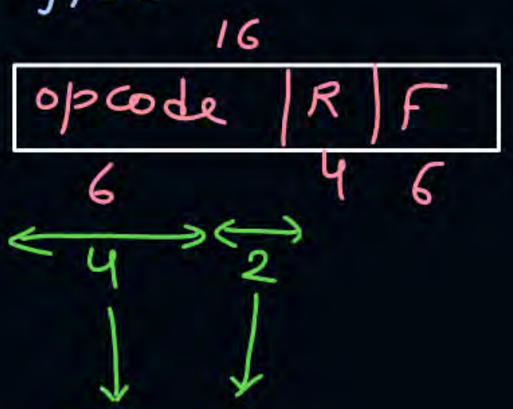


$12 * 2^0 = 12$

used = 8

unused = 4

Type 3

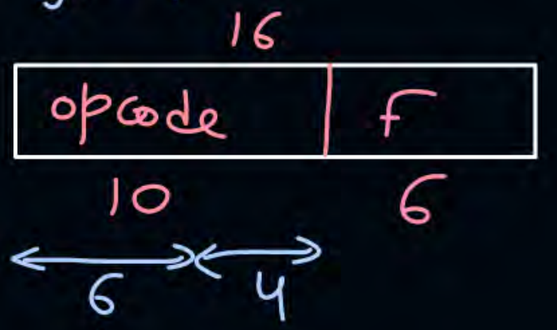


$4 * 2^2 = 16$

used = 14

unused = 2

Type 4



$2 * 2^4 = 2^5 = 32$

Ans.

[NAT]

Ans = 8

#Q. Consider a register-based architecture system (2-address instructions). For this system the following intermediate code is going to be converted in machine code. Minimum number of instructions created are _____?

t1 = X + Y
t2 = t1 * Z
t3 = t2 + M
t4 = Y + t3

Assume X, Y, Z and M are memory operands

t1, t2, t3 and t4 are compiler temporaries

$t_1 = x + y$

$t_2 = t_1 * z$

$t_3 = t_2 + M$

$t_4 = y + t_3$

t3

$R1 = \cancel{x}\ \cancel{t_1}\ \cancel{t_2}$

$R2 = y$

$R3 = \cancel{z}\ M$

Move	R1	X

$R1 \leftarrow X$

Move	R2	Y

$R2 \leftarrow Y$

ADD	R1	R2

$R1 \leftarrow R1 + R2$

Move	R3	Z

$R3 \leftarrow Z$

MUL	R1	R3

$R1 \leftarrow R1 * R3$

MOV	R3	M

$R3 \leftarrow M$

ADD	R1	R3

$R1 \leftarrow R1 + R3$

ADD	R2	R1

$R2 \leftarrow R2 + R1$

[NAT] GATE- PYQ

#Q. In a simplified computer the instructions are:

OP R_i, R_j	- Performs R_i Op R_j and stores the result in R_j
OP m, R_i	- Performs val Op R_i and stores the result in R_i val denotes the content of memory location m
MOV m, R_i	- Moves the content of memory location m to register R_i
MOV R_j, m	- Moves the content of register R_j to memory location m

The computer has only two registers and OP is either ADD or SUB. Consider the following basic block:

t1 = a + b

t2 = c + d

t3 = e – t2

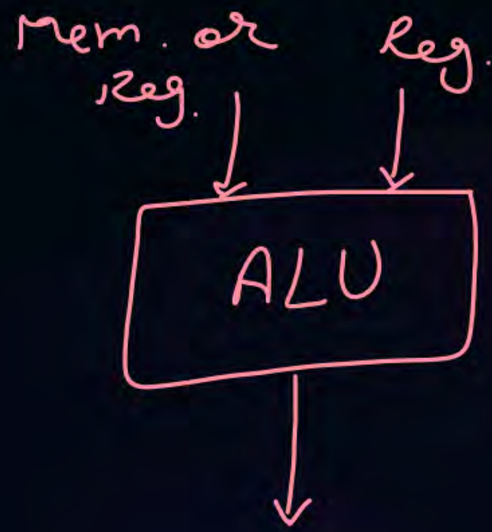
t4 = t1 – t3

R1

[NAT]

Ans = 3

#Q. Assume that all operands are initially in memory. The final value of the computation should be in memory. What is the minimum number of MOV instructions in the code generated for this basic block?



MOV b, R1	R1 ← b
ADD a, R1	R1 ← a + R1
MOV d, R2	R2 ← d
ADD C, R2	R2 ← C + R2
SUB e, R2	R2 ← e − R2
SUB R1, R2	R2 ← R1 − R2
MOV R2, x	x ← R2

R1 = t1

R2 = ~~t2 t3~~ t4

[NAT]

Ans = 5

#Q. In a simplified computer the instructions are:

OP R_i, R_j	- Performs R_i Op R_j and stores the result in R_i
OP R_i, m	- Performs R_i Op val and stores the result in R_i val denotes the content of memory location m
MOV m , R_i	- Moves the content of memory location m to register R_i
MOV R_i, m	- Moves the content of register R_i to memory location m

The computer has only two registers and OP is either ADD or SUB. Consider the following basic block:

t1 = a + b

t2 = c + d

t3 = e – t2

t4 = t1 – t3

[NAT]

#Q. Assume that all operands are initially in memory. The final value of the computation should be in memory. What is the minimum number of MOV instructions in the code generated for this basic block?

$R1 \leftarrow a$

$R1 \leftarrow R1 + b$

$R2 \leftarrow c$

$R2 \leftarrow R2 + d$

$m \leftarrow R2$ — Register spill

$R2 \leftarrow e$

$R2 \leftarrow R2 - m$

$R1 \leftarrow R1 - R2$

$x \leftarrow R1$

$R1 = t1$

$R2 = t3$

$m = t2$

Topic : Register Spill

when enough no. of reg.s are not there in CPU then for a program execution if any operands is moved to memory temporarily.

[NAT]

Ans = 2

#Q. Consider a register-memory architecture system (2-address instructions). For this system the following intermediate code is going to be converted in machine code. Minimum how many registers are required in system so that the code can run without register spill?

Consider first operand is always the register operand and it's the destination for operation too.

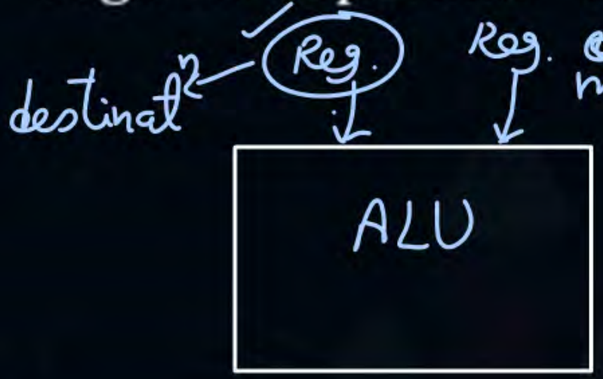
t1 = X + Y

t2 = t1 – Z

t3 = t1 + t2

t4 = M + t3

Assume X, Y, Z and M are memory operands



$R1 \leftarrow X$

$R1 \leftarrow R1 + Y$

$R2 \leftarrow R1$

$R1 \leftarrow R1 - Z$

$R2 \leftarrow R2 + R1$

$R1 \leftarrow M$

$R1 \leftarrow R1 + R2$

$R1 = X + Y + Z + M$

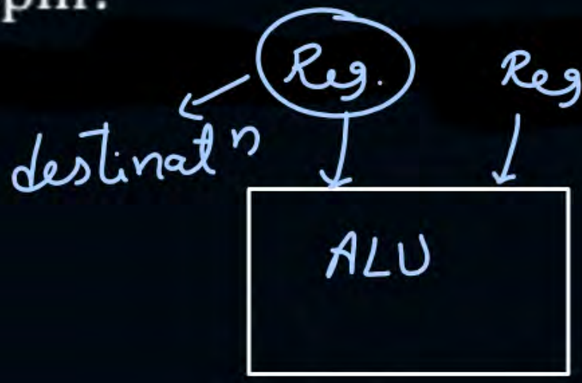
$R2 = t1 + t3$

[NAT]

Ans = 3

#Q. Consider a register-based architecture system (2-address instructions). For this system the following intermediate code is going to be converted in machine code. Minimum how many registers are required in system so that the code can run without register spill?

Consider first operand is [redacted] the destination for operation too.



t1 = X + Y

t2 = t1 – Z

t3 = t1 + t2

t4 = M + t3

Assume X, Y, Z and M are memory operands

$R1 \leftarrow X$

$R2 \leftarrow Y$

$R1 \leftarrow R1 + R2$

$R2 \leftarrow R1$

$R3 \leftarrow Z$

$R1 \leftarrow R1 - R3$

$R2 \leftarrow R2 + R1$

$R1 \leftarrow M$

$R1 \leftarrow R1 + R2$

$R1 = \cancel{X + t1 + t2} \; M$

$R2 = \cancel{Y + t1} + t3$

$R3 = Z$

2 mins Summary

Topic Instruction Construction for CPU

Topic Instruction Cycle

Topic Fetch & Execution Cycle

Topic Branch Instruction

Happy Learning

THANK - YOU