Computer Organisation:

FECHA ___/___/___

Clock cycle = Instruct^n count x cycle per instruct^n

$$\text{CPU Time} = N \times CPI \times T \;/\; \frac{1}{F}$$

Q. A – cycle time = 250 PS CPI = 2.0

B – = 500 PS CPI = 1.2

Instruct^n count — which faster & by how much?

→ $CPU_A = I \times 2.0 \times 250$

$CPU_B = 600\,I \times 1.2 \times 500$

$$\therefore \frac{CPU_B}{CPU_A} = \frac{600}{500} = 1.2$$ ---- A is 1.2 times faster than B.

CPI –

- If diff instruct^n classes take diff numbers of cycle.

$$\text{clock cycle} = \sum_{i=1}^{n} (CPI_i \times \text{Instruction count}_i)$$

- $$CPI = \frac{\text{clock cycle}}{\text{Instruct}^n \text{ count}} = \underbrace{\sum_{i=1} \left(CPI_i \times \frac{\text{Instruct}^n \text{ count}_i}{\text{Instruct}^n \text{ count}} \right)}_{\text{Relative frequency}}$$

Kai Hwang

Q Alternative compiled code sequences using instruct^n in class A, B, C

class	A	B	C
CPI for class	1	2	3
IC in seq 1	2	1	2
IC in seq 2	4	1	1

① Seq 1 IC = 5 seq 2 IC = 6

- Clock cycles
  $2\times1 + 2\times1 + 3\times2 = 10$
- clk cycle
  $4\times1 + 2\times1 + 3\times1 =$
- Avg CPI = 2.0

Q 1 A 400 MHz processor was used to execute a benchmark program with following instruct^n mix & clock cycle counts:

Instruct^n type	count	clk cycle count
Int arithematic	450000	1
Data Transfer	320000	2
Floating	150000	2
Control Transfer	80000	2

CPI, MIPS rate & execution time?

$$\frac{450000\times1 + 320000\times2 + 150000\times2 + 80000\times2}{(450000 + 320000 + 150000 + 80000)}$$

Avg CPI = 1.55 ---- ①

② Execution Time: $N \times CPI \times T$

$$= \frac{N(IC) \times CPI}{f}$$

$$= \frac{\times 1.55}{400\times10^6} = 3.87 \text{ ms}$$

Instruct^n type	CPI	Instruct^n mix
Arithematic	1	60%
store with cache hit	2	18%
Branch ~~Memoney eg~~	4	12%
Memory eg	8	10%

CPI = Avg CPI = 2.24 $(1\times0.6 + 2\times0.18 + 4\times0.12 + 8\times0.1)$

$$MIPS = \frac{400\times10^6}{2.24\times10^6} = 175.57 \text{ mIPS}$$

MIPS Rate:

$$MIPS = \frac{Ic}{T\times10^6} = \frac{f}{CPI\times10^6} = \frac{f\times Ic}{C\times10^6}$$

$$\frac{400\times10^6}{1.55\times10^6} = \frac{8000}{155} = \frac{400000}{31} = 258 \text{ million inst per second (MIPS)}$$

John L David A Patterson

1.6

Q The execution time of 4 prog on Three comp
Assume 10^9 instruction were exe

Exe Time (sec)

prog	A	B	C
1	1	10	20
2	1000	100	20
3	500	1000	50
4	100	800	100

→ Calculate MIPS of each & plot on graph

Amdahls law - fixed load Model: defines

$$\text{Execution time} = \text{Exe time}_{ow}\left[\left(1 - \text{fraction}_{enhanced}\right)\frac{f}{S}\right]$$

$\text{fraction}_{inhance} \leq 1$ (f)

$\text{Speedup}_{enhance} > 1$ (s)

$$S = \frac{1}{(1-f) + 1/s}$$

CA TUT 25-17

Q.1

Fractionenhance = 0.4 speedupenhance = 10

$$\text{speedupoverall} = \frac{1}{0.6 \times \frac{0.4}{10}} \approx 1.56$$

Q.2 → for FP case: S = 1.6, f = 0.5

for FPSQR: f = 0.2, S = 10

$$\therefore \text{overall speedup} = \frac{1}{(1-f) + f/S}$$

$$= \frac{1}{(1-0.5) + 0.5/1.6} \quad \text{---- for FP}$$

$$= 1.23$$

Same for FPSQR = 1.22

overall speedup of FP is greater hence choose FP.

Q.3 → FP op = 25%
Avg CPI = 4.0
CPI of other = 1.33
freq of FPSQR = 2%
CPI of FPSQR = 20

Solve by:

$$CI = N * CPI * I \cdots \text{processor performance equation}$$

$$= \frac{N * CPI}{f}$$

→ CPI with new FPSQR = CPI original − 2% (CPI old FPSQR − CPI of new FPSQR only)

$= 2.0 \times 2\% (20 - 2) = 1.64$

we can compute CPI of enhancement of all FP instructions the same way or by summing the FP and

Speedup new FP = $\frac{\text{CPU Time Original}}{\text{CPU Time new FP}}$

$= \frac{IC \times Clk\,cycle \times CPI\ Org}{IC \times clock\,cycle \times CPI_{new}}$

Q.4 I dont understand ~~interstellar~~ whats going on.

(a) - 1.25 (b) → $\frac{1}{1-0.99+\frac{0.99}{2}} = 1.98$ (2 → speedup)

(c) 1.19 (d) 1.11

FECHA 29 July 2024

Q.6

a) when parallelizing an app, ideal speedup is speeding up by the number of processor. This is limited (why I am not understanding this)

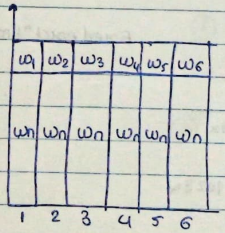
Q.7 Speedup from pipelining

$= \frac{\text{Avg Instruct}^n \text{ time unpipelined}}{\text{Avg Instruct}^n \text{ time pipelined}}$

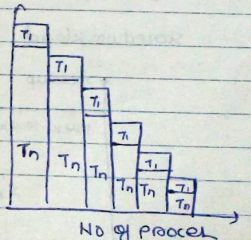
$= \frac{4.4ns}{1.2ns} = 3.7 \text{ times}$

Amdahl's law:

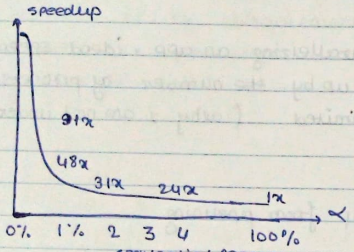
$$S_n = \frac{n}{1+(n-1)\alpha}$$



Fixed

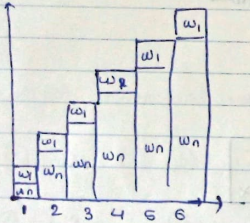


Decreasing Exect^n time

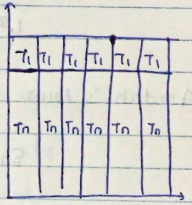


Speedup with fixed load

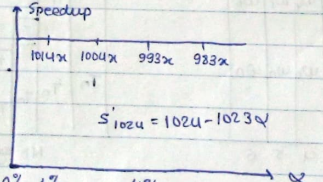
Gustafson's law



Scaled workload



Fixed exct° time



Speedup with fixed

200 Km/h

$$Sn = \frac{w_i + w_n}{w_i + w_n/n} = \frac{w_1 + G(n)w_n}{w_1 + G(n)w_n/n}$$

Amdahl & Gustafson are special case of fixed memory (memory bounded speedup)

Instructions

3 add
2 add
1 -.-
0 -.-

$Z = K + B * C$

OP —, —, — (3)
d
operands

d - dest
S - source

OP —, — (2)
d↑ S↑

① OP — — add contents with Acc
S↑ store in Acc.

⓪ OP↑

FECHA 5/Aug

3 Address : ~~X+B*C~~ 1 Add :

~~2 Address~~ :

MPY Z, B, C
ADD Z, Z, K (2 Instruct°)

2 Address :

MPY B, C	MOV R1, C
ADD K, B	MUL R1, B
MOV Z, B	ADD R1, K
	MOV Z, R1

(4)

1 Address :

LOAD C
MUL B
ADD K
STORE Z

(4)

Postfix of Z = K+B*C = KBC*+

0 Add - PUSH K
PUSH B
PUSH C
MUL
ADD
POP

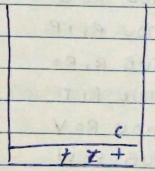
(6)

FECHA

$Y = \frac{A-B}{C+(D \times E)}$

3-Address :	2 Add	1 Add
SUB Y A B	MOV Y, A	LOAD D AC ← D
MPY T, D, E	SUB Y, B	MPY E
ADD T, T, C	MOV T, D	ADD C
DIV Y, Y, T	MPY T, E	STORE Y Y ← AC
	ADD T, C	LOAD A
	DIV Y, T	SUB B
		DIV Y
		STOR Y Y ← AC

(A-B) / C + (D * E)



AB - C / DE * + is the postfix

PUSH A
PUSH B
SUB
PUSH C
DIV
PUSH D , PUSH E
MUL
ADD POP

10 Instruct°

$$z = \left(\left((P-(Q+S)*T)\right) \Big/ \left(U/(V-W)\right)\right)$$

3dd →

```
SUB   R1, V, W
DIV   R1, U, R1
ADD   R2, Q, S        (6)
MUL   R2, R1, T
SUB   R2, P, R2
DIV   Z, R1, R2
```

2 Add

```
~~LOD~~  MOV   R1, W
      SUB   R1, V
      DIV   R1, U
      MOV   R2, S
      ADD   ~~R2~~ R2, Q        (9)
      MUL   R2, T          6-11
      SUB   R2, P
      DIV   R2, R1
      MOV   ~~R~~Z, R2
```

SIR

```
MOV R0, S
ADD R0 Q
MOV R1, P
SUB R1 R0
MUL R1 T
MOV R0 V
SUB R0 W
MOV R2 U
DIV R2 R0
DIV R1 R2
MOV Z R1
```

1 Address (16)

```
LOD W
SUB V
DIV U
STOR U
```

```
LOD Q
ADD S
STOR T
LOD P
SUB T
```

0 Add

1 Add - ≡15

```
LOD Q
ADD S
STORE T        MUL T
LD P           STORT
SUB T
MUL T
LOD V
SUB W
STORE T1
LD U
DIV T1
STORE T1
LOD T
DIV T1
STORE Z.
```

PQS+T*-UUW-// (14)

```
PUSH P
     Q
     S
ADD
```

- Fixed length, variable length instruction.

R7
8 4 2 1

OP operand

1 2 3

16 bits & 16 R

then no number of type 3 addl instruct°

30 type 2

28 type 1

64 type 0

type 3:

each operand field 3 bits:

∴ 4 × 3 = 12 bits for operands.

0000 01, 02, 03

1101

Type 2: 2 operands ∴ 2 × 4 = 8 remaining 8 opcode

1110 0000

111 1101

Type 1 1 operands ∴ 1 × 4 = 4 remaining 12 opcode.

0000

1111111 00000

(27)

Type 0: All 16 bits represents op

1111111111 00 0000

1111111111 11 1111

By solving these all to get total no. of add

Type 3 -14×2^{12} 3

$+ 80 \times 2^{8}$ Type 2

$+ 28 \times 2^{4}$ Type 1

$+ 64$

655 $\equiv 2^{16}$

Example 1:

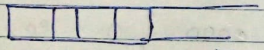
Consider a machine with 16 bit instruct° & 16 reg.

Instruct° format:

opcode + Mem address

→ • If 48 KB byte addressable mem · we need 12 bit to specify addl locat°

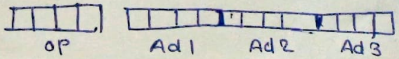
• Remaining for 4 bits opcode



Opcode + Reg Add

need 4 bits to select one of the 16 available reg.

Suppose we have 4 bit op encode 16 diff ins- with 3 operand 3 × 4 = 12 bit



FECHA

Ex. 2 Consider machine with 16 bit instruct^n & 16 registers. And wish to encode the following instruct^n.

- 15 instruct^n with 3 Add

14	2
31	1
16	0

Can we encode this instruct^n set in 16 bits?

→ Yes, if we can use expanding opcodes.

⇒ 0000 ---- ---------- } 3

14 − 1110

escape code — 1111 0000 R1 R2 } 2

1111 1101 (13) R1 R2

7 masked 1111 1110 0000 } 1

1111 (1 1110)-(31)

1111 1111 1111 0000 } 0

-/-/- 1111

15 instr

+ $15 \times 2^4 \times 2^4 \times 2^4 = 15 \times 2^{12}$ = 61440

14 Ins

14×2^8 = 3584

31 ins

31×2^4 = 496

16 instr

16×2^0 = 16

65536 ≡ 2^{16}

we have exact match with no wastage Hence possible.

Ex. 3 Is it possible to design an expanding opcode to allow the following to be encoded with 12 bit ? Assume reg operand required 3 bits

4 instr with 3 reg :: 0000 R1 R2 R3

255 instr with 1 reg 0000 - -

16 instr with 0 reg

→ $4 \times 2^9 + 255 \times 2^3 + 16 = 4104$ → not po

instruct^n leng 12 bit ∴ $2^{12} = 4096$

000 R1 R2 R3 || 100 000 000 --- } TYPE 1

100 -/-/- —

111 111 111 0000

111 111 111 1111

1111 1111

32 16 8 4 2 1

Ans 64

Q4 Assume CPU having 16 reg & instruct^n length is 16 bits. It should be having 14 of type 3 add instruc

31 Type2

12 Type1

Calculate almost how many no. of Type 0 instruct^n should have. Show proper encoding.

Type 3: 0000 R1 R2 R3

1101 R1 R2 R3

Type 2: 1110 0000 R1 R2

1111 1110

Type 1: 1111 1111 0000 R1

1111 1111 1011 R1

Type 0: 1111 1111 11DD 0000

1111 1111 1111 1000

$14 \times 2^{12} + 31 \times 2^{8} + 12 \times 2^{4} + x2^{0} = 2^{16}$

williamsten

- Control unit functional Requirement

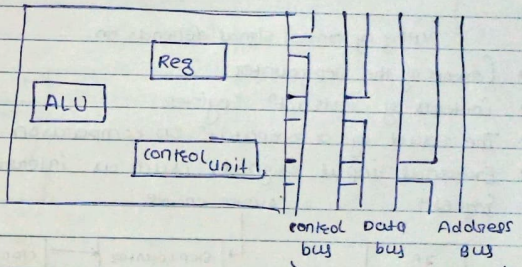
Control unit → generate control signal

It peyorms basic task:

① sequence of micro-operat^n

② Execut^n

496 chapter 14 21 chapter:



control signals

External ALU

- Datapth
- Control path

© when stores unstores is ALU involved?

chapter 20 716

Q Who generate Addre?

→ Processor

FECHA

MK morgan kaufmann

RISC & NSC pattern book

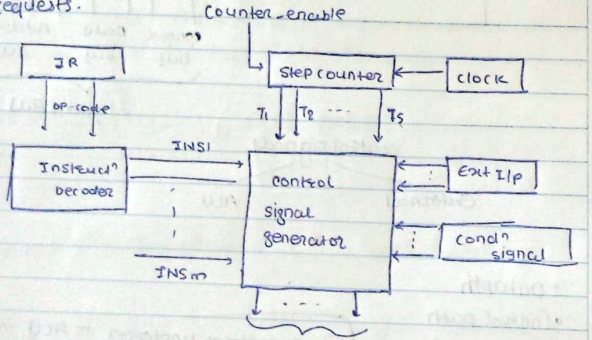
2 Approaches to generate

Hardwired

Microprogramed control.

Setting of control signal depends on.

- Contents of the step counter
- Contents of instruct° register
- The result of a computat° or comparision operat°
- External input signal, such as interrupt requests.



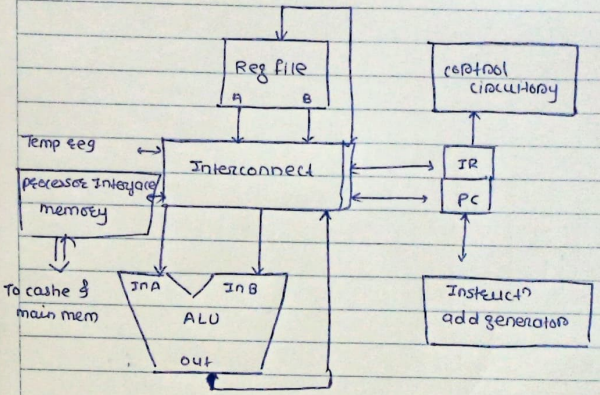
Generat° of control signa

FECHA

Datapath control signal:

RF_write = TS. (ALU + load + call)

CISC cycle processors:



organisat° of cisc-cycle processor.