Arc, ক্লাস-1

Overview

ISA → instruction Set Architecture.

~~DOP~~
fact → যেটা যেইরকমই থাকবে সত্য।
Theory → fact কে explain করার জন্য যে পদ্ধতি অবলম্বন করি

Classes of Computer

(I) Personal Mobile Device, e.g: Smart phone

(II) Desktop.

↓ সময়ের বিভিন্নতার উপর comp. Arch Depend করে।

- ISA
- Microarchitecture
- Register-Transfer level (mp related)

} c: Arc এর topics ↑

Slide-19
Book-2 থেকে

~~DOP~~
Parallelism →
- DLP
  ~~Memory~~ অনেক গুলো Data at a time নিয়ে কাজ করা।
- TLP
  ~~এক~~ Task নিয়ে কাজ করে।

· Parallelism এর 4 major point } Task level.

Scanned by CamScanner

Arch. vs Micro Arch

Arch. → also known as ISA

Micro Arch → also known as Organization

Why we need Arch?

(1) Improve Performence & (ii) Abilities. → ব্যবহার specialized ফিচার হবে

Arch. এ আমরা future trend নিয়ে চিন্তা করতে হয়, চিন্তা করে আসতে হয়।
বর্তমান বাস্তব চিন্তা করে বানালে obsolete comp.
future এ বসে চিন্তা করে আমরা future comp

Moore's law

Memory র capacity যে হারে বাড়ছে, Processor এর capacity সে হারে বাড়ছে না,

IRAM 29

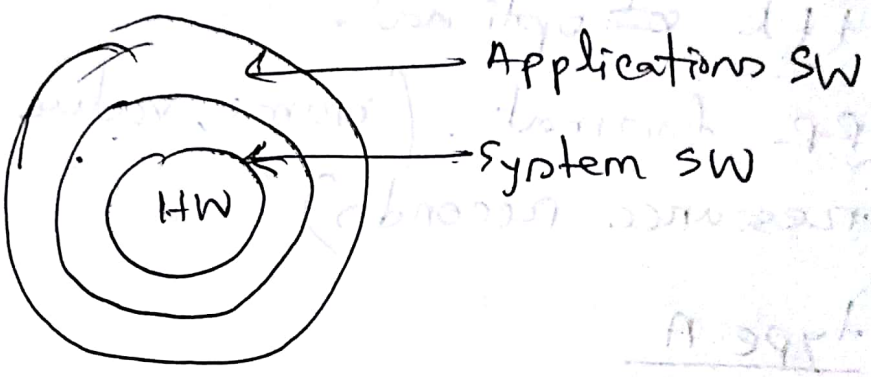
Abstraction → user থেকে hide করা

Date : ___/___/___

Arch, C-3

Computer Organization & design

Internal design of computer



System SW

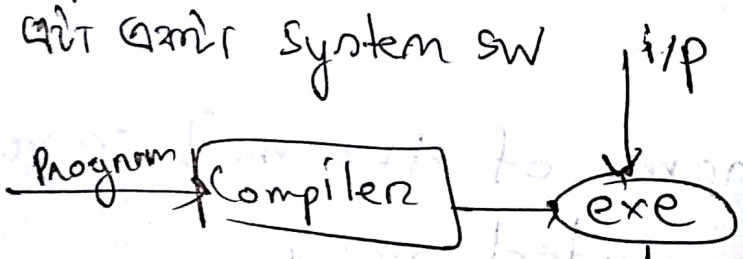
Basic difference between Application SW & system

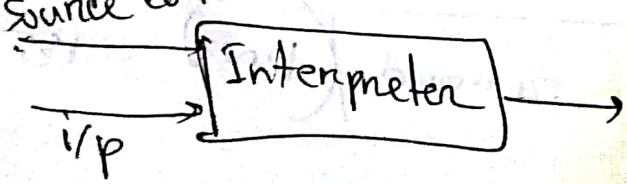
OS: HW resource-গুলোকে effective manage করে

1. Processor
2. Memory
3. I/O

Compiler

এটি একটি System SW





Protivas

একের কাছ একটা lang. কের অন্যের lang ও convert করা (translaton)

Natural Lang
High-level lang

↓

Compiler

↓

Assembly lang

↓

Assembler

↓

Machine lang

Computer এর component

5 টি : (I) I/P (II) O/P
Ex: keyboard, Mouse. Direction: PC কের device ও ex: Projector

(III) Memory (IV) Data path

Control
one kind of signal

(I) Processor
(II) Memory
} এরা device computer এর basic criteria.

Date : ____/____/____

ISA (Instruction set Architecture)

~~System Sw~~

Application Binary Interface (ABI)

Application program & OS এর মধ্যে layer

☹ may be

12

17

equations

Networking C-9

FTP

File Transfer Protocol.

Client - Server থেকে file নেয়া / দেয়া,

Server -

FTP client - local file system

→ client - server model.

→ Server port : 21.

(establish standard Protocol)

port use করে

Protivas

Arch, c-4

CPU Time = I. count × CPI × Clock cycle Time

$$\text{CPU Time} = \frac{\text{I. count} \times \text{CPI}}{\text{clock Rate}}$$

$$\text{CPU clock cycles} = \sum_{i=1}^{n} (CPI_i \times C_i)$$

CPU clock cycles$_1$ = 2×1 + 1×2 + 2×3
= 10 cycles

CPU clock cycles$_2$ = 4×1 + 1×2 + 1×3
= 9 cycles

	A	B	C
CPI	1	2	3

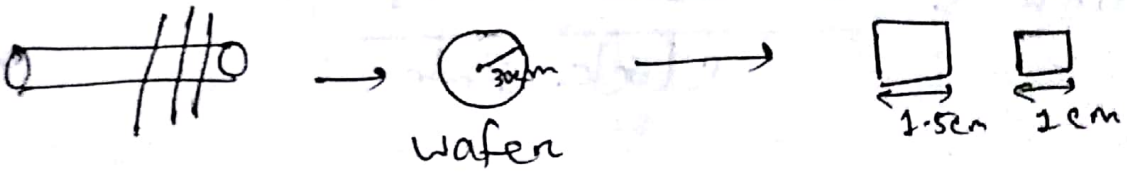
Clock s.	A	B	C
1	2	1	2
2	4	1	1

CPI : Clock cycle Per Instruction

Wafer - die related Math:

Wafer 30 cm ~~radius~~ diameter.

die side : 1.5cm / 1.0 cm.



$$\#\text{die per wafer} = \frac{\text{wafer Area}}{\text{Die Area}} - \frac{\pi \times \text{wafer Diameter}}{\sqrt{2 \times \text{Die Area}}}$$

So, Dies per wafer $= \frac{\pi \times \left(\frac{30}{2}\right)^2}{(1.5)^2} - \frac{\pi \times 30}{\sqrt{2 \times 2.25}}$

$\therefore = 270$

For 1 cm:

$$\text{Dies per wafer} = \frac{\pi \times \left(\frac{30}{2}\right)^2}{1^2} - \frac{\pi \times 30}{\sqrt{2 \times 1}}$$

$= 640.$

no. of die বাড়লে ↓ সাধারণ cost per die কমে যাবে।

Home Assignment

→ Chip → How it works / manufactured

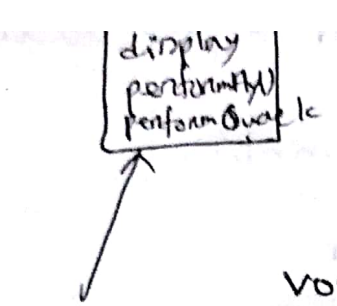
→ Memory →

Processor →

(বই থেকে chapter 1 পড়ে ফেলা)

Hard Drive vs SSD

USB-B vs USB-C




```
}.

void performFly() throws CantFly{
    if (flyBvr == null)
        throw new CantFly();
    flyBvr.fly();
}
```

@ Arch_ C-5

CHAPTER 2: (COD: HSI)

Instruction: the language of the computer

ISA: Instruction Set Arch

a = b + c instruction টার corresponding binary value কে কীভাবে translate করে? via ISA.

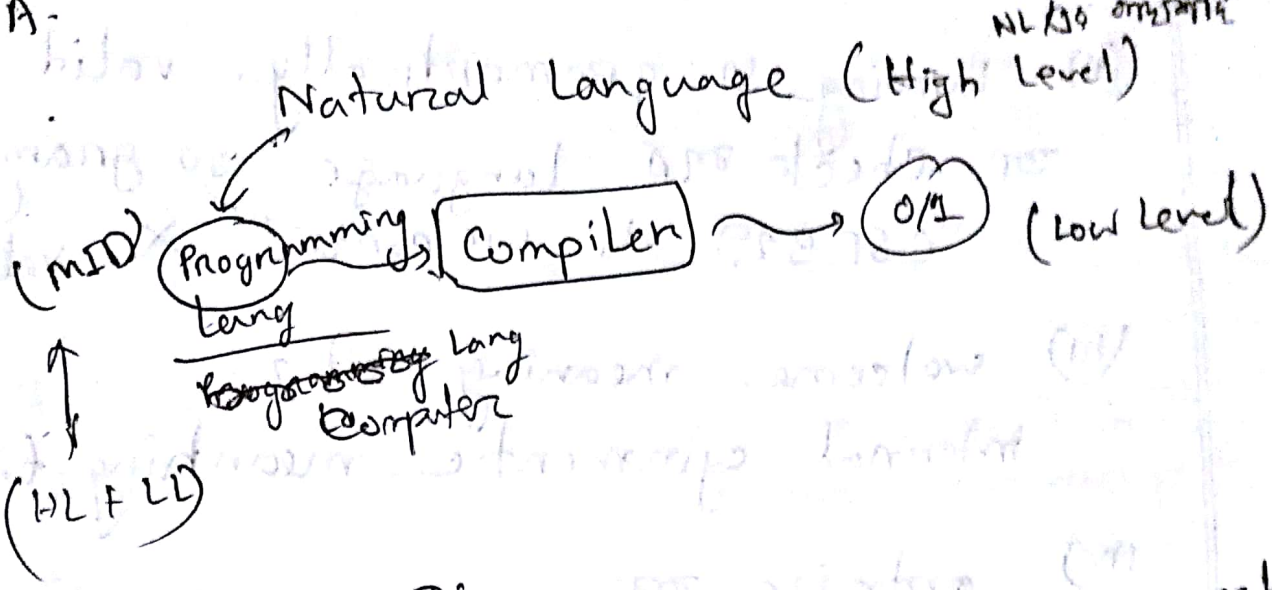
Protivas

Date / /

Compiler Interpreter

MIPS ISA. Millions of Instruction Per Second.

Computer এ যে instruction গুলা বা কমান্ড binary তে রূপান্তর হয়ে এর internal architecture. একে ISA.



Compiler এর কাজ:

Source Lang থেকে কোডগুলো Lang এ convert করা। যেমন: C Lang কে Assembly তে রূপান্তর করা হয়।

Compiler Phase Steps:

i = i + 5 →
add $s0, $s0, #5

(I) lexical Analysis
(II) Parsing
(III) Semantic Analysis
(IV) Optimization
(V) Code Generation

(I) character by character পড়ে meaningful word তৈরি করি। (word identify করার) এই word কে Compiler এর ভাষায় token বলা হয়।

```
a = b+c;
↑ ↑ ↑ ↑ ↑ ↑
tokens.
```

(II) Parsing এ 'gramartically valid কিনা' তা check করে। Language এর grammar "CSE 329 to welcome" ✗ Grammar valid কিনা

(III) welcome meaning কী?
Internal symmentic meaning find করা

(IV) optimize করা,

```
b=0
a=b*c;
|b=0
|a=0   } multiplication optimize করলাম
```

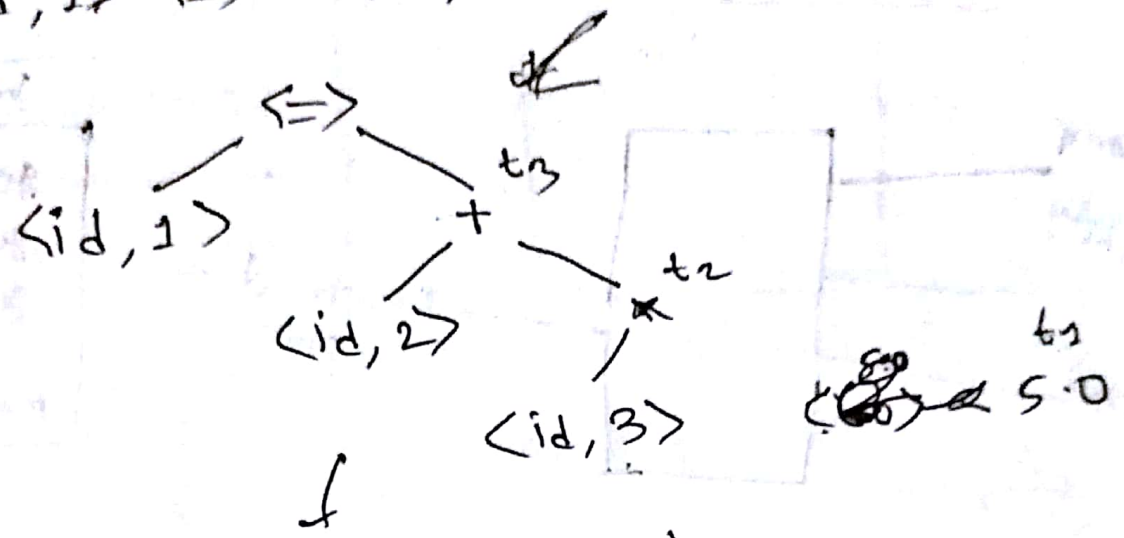
unused variable optimize করা,

(V) Code generate করা,

a = b + c * 5.0

(I) <id,1> <=> <id,2> <+> <id,3> <*> <5.0>

(II)



(III)
t1 = int to float (5.0)
t2 = id3 * t1
t3 = id2 + t2
id1 = t3.

} intermediate code generation (symmentic analysis part)

↓

(IV)
t1 = id3 * 5.0
id1 = id2 + t1

} optimized

↓

LD R2, id3 (load)
Mult R2, R2, #5.0 (multiplication)
LD R1, id2 (load)
add R1, R1, R2 (add)
st id1, R1 (soln)

Date: / /

Arch, c-6

chapter-2 ; class-2

Instruction Set:

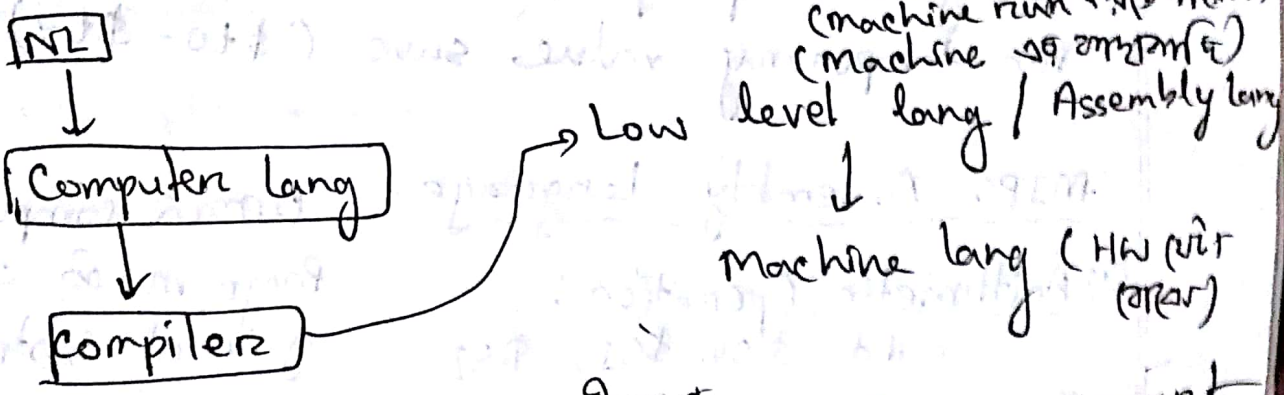
The vocabulary (binary string) of commands understood by a given Architecture.

Example:

(i) MIPS (ii) ARM (iii) X86 (iv) RISC.

Stored Program Concept:

The idea that instruction and data of many types can be stored in memory as numbers.



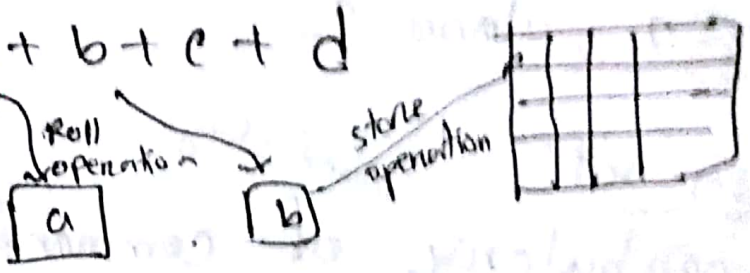
→ প্রতিটা instruction কে binary string এ convert করতে হবে তাই define করে ISA.

→ ISA এর concept এসেছে stored program concept থেকে।

→ ISA এর উপর depend করেই HW Design করতে হয়।

এক এরকম ISA এর উদাহরণ HW এ প্রদর্শন করা হয়,

$a = a + b + c + d$



② MIPS Operand

32 Reg:

$ s0 - $ s7, $ t0 - $t9, $zero, $a0 - a3,
$ v0 - $v1, $gp, $fp, $sp, $ra, $at.

(32 টি register আছে)

for general purpose ($ s0 - $7)
for temporary value save. ($t0 - $t7)

MIPS Assembly Language:

(1) Arithmetic Operation:

```
add  $s0, $s1, $s2.
sub  $s0, $s1, $s2
addi $s0, $s1, 40.
```

(2য়ার computer program এ 5 টি category general instruction use করা.

(1) Data Transfer:

```
lw  $s1, 20($s2)
sw  $s1, 20($s2)
```

load word
store word.

Date ___/___/___

(iii) Logical operation:

and $s1, $s2, $s3

or

nor

Conditional Branch:

beq $s1, $s2, 25

bne $s1, $2, 25

slt $1, $2, $3.

Unconditional Jump:

Principle of designing ISA:

(i) Simple Design

(ii) Smaller and faster.

$zero register এ সবসময় 0 থাকে

mov $s1, $s2 } same instruction

add $s1, $s2, $zero

$$= \frac{1}{2} \cdot \frac{s^2 - 2s + 4}{(s+1)^2 (s-2)}$$

$$= \frac{1}{2} \frac{s^2 - 2s + 4}{2(s+1)^2 (s-2)}$$

Arch. C-7

ISA

MIPS Assembly language ↳ 5 টি category র instruc রয়েছে।

MIPS A2 ↳ convert করলে

a = b + c	add a, b, c
d = a - e	sub d, a, e

} instructions পাবো

কিন্তু যদি আমরা register level ↳ চিন্তা করি, তাহলে variable গুলোকে register ↳ load করতে হবে প্রথমে. ধরি,

a → $s0
b → $s1
c → $s2

} load করা লাগবে, then → add $s0, $s1, $s2

sub $s3, $s0, $s4.

d → $s3
a → $s0, e → $s4.

instruction : $f = (g+h) - (i+j)$

$f \to \$s0$, $g \to \$s1$, $h \to \$s2$, $i \to \$s3$, $j \to \$s4$

design principle
- Simple is better
- As smaller as faster

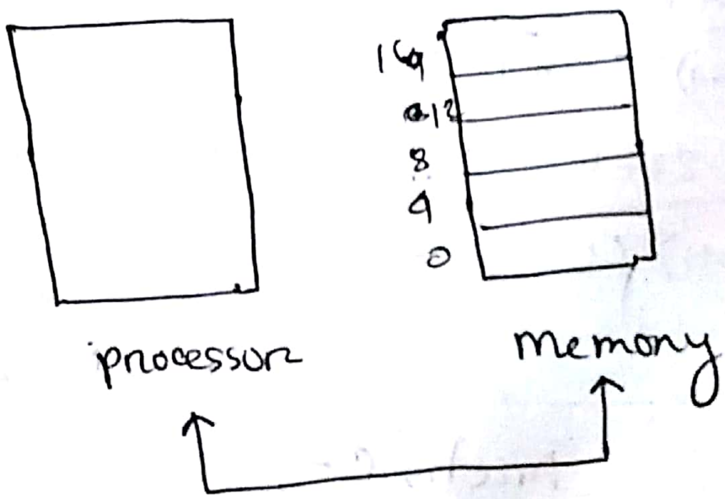
MIPS AL :

add $t0, $s1, $s2

add $t1, $s3, $s4

sub $s0, $t0, $t1.

Data Transfer :



MIPS এ 2টি command আছে।

lw : memory থেকে word register এর মধ্যে আনে

sw : register থেকে memoryর মধ্যে আনে

1 word = 32 bit = 4 byte.

$g = h + A[8]$ ($g \to \$s1$, $h \to \$s2$, $A \to \$s3$ (base Address))

lw $t0, 8($s3)

অথবা, (8×4)

lw $t0, 32($s3)

base : $s3

8th index : 8($s3).

MIPS এর মেমরি Address গুলো step 4 bit করে।

Scanned by CamScanner

Binary → Decimal:

1 0 1 1 | $d \times (base)^i$

$= 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$

$= 8 + 0 + 2 + 1$

$= 11.$

| . . . | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

most sign   least significant bit

unsigned হলে bit সংখ্যা possible 2^{32}.

$(x_{31} \times 2^{31}) + (x_{30} \times 2^{30}) + \dots + (x_0 \times 2^0) \Rightarrow$ যেকোন সংখ্যাকে binary তে রূপান্তর।

negative সংখ্যা হলে -

$(-x_{31} \times 2^{31}) + (x_{30} \times 2^{30}) + \dots + (x_0 \times 2^0) \Rightarrow$

1111 1111 1111 1111 1111 1111 1111 1100 = -4

x_{31} x_0

$(-x_{31} \times 2^{31}) + (x_{30} \times 2^{30}) + \dots + (x_0 \times 2^0) = ?$

0000 0000 0000 0000 0000 0000 0000 0010 = 2

1111 1111 1111 1111 1111 1111 1111 1101 = (Negtv করে) (1's comp)

1111 1111 1111 1111 1111 1111 1111 1111

+1

add $t0, $s1, $s2 , encode করতে হবে 32 bit এ। যাতে ঐ binary code দেখলেই বুঝি ~~আসলে~~ main code এ add করতে বলেছে।

	shift	5	5	5	6bit

opcode 6 bit — register এর জন্য

32 টা register so, 5 bit লাগে।

0	17	18	8	0	32
add	$s1	$s2	$t0		
000000	10001	10010	01000	00000	100000

← MIPS code এ convert করলে এটা আসে

Finally memory তে input যাবে এই 01 instruction গুলো

copy করা অর্থ 0 register এর মান যোগ করা।

2 bit এর কেবল 2 টা Number যোগ করব,

যেকোন instruction set 32 bit দিয়ে represent করা : basic concept of MIPS Arch.

op	rs	rt	rd	shamt	funct
6	5	5	5	5	6

rs = source operand, প্রথম register

rt = 2nd register.

rd = Register description

shamt = shift amount, funct = function

Date: / /

function define করা থাকলে তাকে mean বলে,

(So, $t_0 = 8$
$= 16$)

Laplace, C-

$\Rightarrow$ If $f(x) = \frac{a}{2} + \sum_{n=1}^{\infty} (a_n e_0 \cos nx + b_n \sin nx)$, $-\pi <$

then prove that (i) $a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\,dx$

(ii) $a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos nx$

(iii) $b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin nx$

Prove:

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos nx + b_n \sin nx) \quad \text{—(1)}$$

multiplying both sides of (1) by dx and integra

this from $n = \pi$ to $n = -\pi$, we get,

Date : / /

Arch, C-8

A[300] = h + A[300]

↳ t1 ↳ $s2

4 byte me offset
so, 300 × 4 = 1200

loading

lw $t0, 1200($t1)

add $t0, $s2, $t0,

sw $t0, 1200($t1)

storing

6bit	5	5	5	5	5	= 32 bit.
op	rs	rt	rd	address/shamt	funct	
35	9	8	1200 (16 bit			
0	18	8	8	0	32	
43	9	8	1200			

MIPS Architecture

Date : ____/____/____

R-type operation হলে (11) no. এ যাবে.

MIPS ~~Code~~ ই হবে,
Architecture ই হবে

MIPS Arch এ I,J-type এ instruction/operand এ মধ্যে 16 bit দিয়ে instruction info. দেয়, আর 16 bit দিয়ে Address কে indicate করে.

for (1), MIPS machine code:

1000 11 01001 01000 0000 0100101 10000

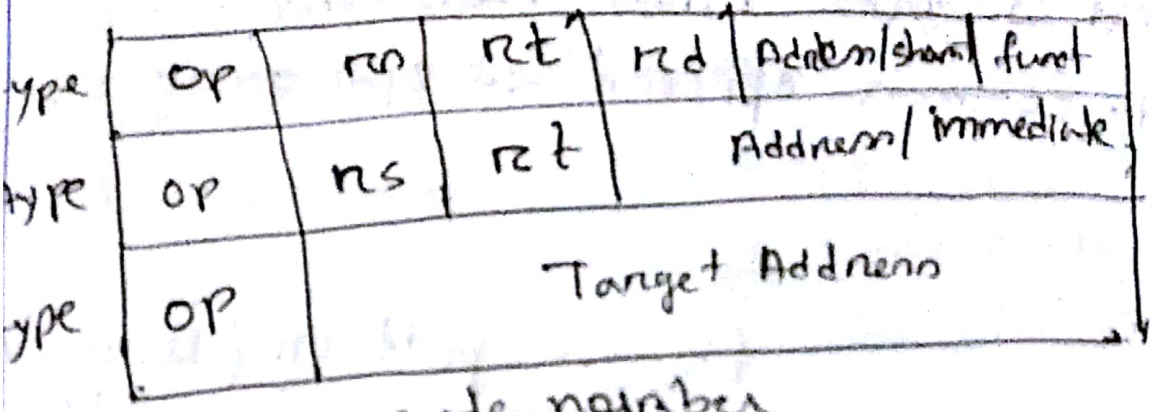
00000 10010 01 0000 1 000 000 00 100000

10101101001 01000 000 00100 10 11 0000

* S_0 - 16 দিয়ে
* t_0 = 8 দিয়ে
* শেষ 6 bit দেখে decide করতে হবে এটা কোন type instruction

Exam ✓ A[300] = h + A[300] ⟷ machine code এ convert করতে হবে

Date : / /



code number

$zero → 0
$at → 1 (reserved, we could not use it. Assembler use this)
$v0 - $v1 → 2-3
$a0 - $a3 → 4-7
$t0 - $t7 → 8-15
$s0 - $s7 → 16-23 (Saved) → saved value use ...
$t8 - $t9 → 24-25
$k0 - $k1 → 26-27 → special purpose
$gp → 28 → global pointer
$sp → 29 → stack "
$fp → 30 → frame "
$ra → 31 → returned Approach

Arch, c-9

Logical Operant:

Shift Left logical (sll)

Shift Right logical (srl)

sll $t2, $s0, 4 . s0 এর 4-2^4 bit left shift করবে,

op	rs	rt	rd	shamt	funct
0	0	16	10	4	0

(operation এর mode define করে)

32 বারকম Normal Add

~~op~~ I type

→ operation type এর উপর ভিত্তি করে maximum possible address কত declare করা যাবে এগুলো 26 bit থাকে

Hexadecimal: যেকোন Architecture এ লিখতে হবে,

00 8f --- (hex)

0000 0000

১২৮ bit এর value হলে এটা দেখে decision নিবে কোন type operation.

and $t0, $t1, $t2.

or

nor2

Now $s3, $s4.

```
if (i==j)
{
    f = g + h
        1
    ↓   ↓   ↓
   $s0 $s1 $s2
}
else
{
    f = g - h;
}
```

MIPS Assembly code:

```
(i)   bne $s3, $s4, Else
(ii)  add $s0, $s1, $s2
(iii) j Exit
(iv)  Else: sub $s0, $s1, $s2
(v)   Exit:
```

(1) branch not equal, $s3, $s4 Else → unconditional jump

loop:

```
While (save[i] == k)
      $s6  $s3  $s5
{
    ≡
    i += 1
}
```

```
Loop: sll $t1, $s3, 2
      add $t1, $t1, $s6
      lw  $t0, 0($t1)
      bne $t0, $s5, Exit
      add $s3, $s3, 1
      j loop
Exit:
```

Base Address of save stored in $s6.

(i) $t1 stores $s3 কে 2^2 bit shift করলে (এর Add কে) যে Address পাওয়া যায়।

(ii) $t1 এ offset address আছে। এর সাথে $s6 এর Address যা (base) যোগ করলে actual physical address store হবে যা loop $t1 এর মধ্যে। save[i]

(iii) $t1 এ save[i] এর value আছে, Address অনুযায়ী এখানে, OFFSET 0, কারণ, target address দেওয়া আছে।

t1 address এ যে value আছে তা load করে t0 তে।

lw reg value (reg)

- reg ← এখানে পাঠাবে
- value → এই Address আছে → offset হবে।
- (reg) → এর value কে base address ধরে এর মধ্যে যাবে।

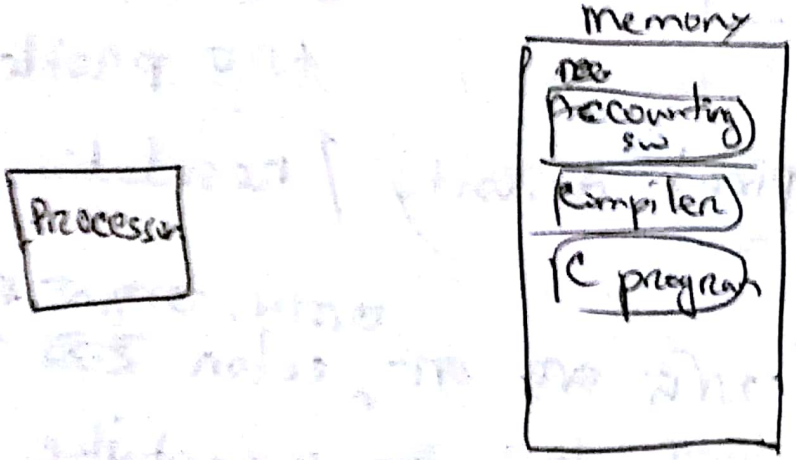
C code → MIPS Assembly code → Numeric value
↓
0 1

Date : / /

- Today computers are build on Two key principles:

(I) Instructions are represented as Numbers.

(II) Programs are stored in memory to be read or written, just like data.



Date : ____ / ____ / ____

Arch, C-10

MIPS Assembly Lang

- Jump and Link

  most used: jal proceedureAddress

- Jump Register

  jr $ra — special register, return adress

- Jump

  j address.

- $a0 - $a3 ⇒ Parameters.
- $v0 - $v1 ⇒ Return Values.
- $ra ⇒ Return Address.
- $sp ⇒ Stack Pointer

```
int Leaf_example (int g, int h, int i, int j)
{
    int f;
    f = (g+h) - (i+j)
    return f;
}
```

(g → $a0, h → $a1, i → $a2, j → $a3)

procedure in C যেভাবে লেখে। MIPS Arch এ convert করে।

Protivas

Scanned by CamScanner

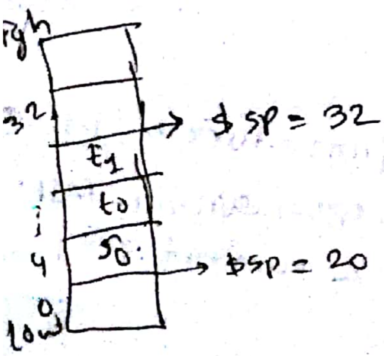
MIPs Arch

Leaf_example: (label)

```
addi $sp, $sp, -12
sw   $t1, 8($sp)
sw   $t0, 4($sp)
sw   $s0, 0($sp)

add  $t0, $a0, $a1
add  $t1, $a2, $a3
sub  $s0, $t0, $t1

add  $v0, $s0, $zero
lw   $s0, 0($sp)
lw   $t0, 4($sp)
lw   $t1, 8($sp)

addi $sp, $sp, 12

jr   $ra
```

(sw lines:) t_1, t_0, s_0 তে যেসব আগের value store করছি, সেই গুলোর জন্য place এ

(lw lines:) load করলাম previous value আবার।



যদি 12 বিয়োগ করি, অর্থাৎ 3 টা address নিচে আসবে $sp = 20.

যেহেতু, return এর value v_0 তে store হয়, v_0 তে value থাকলেই function return এর কাজটা হয়ে যাবে।

Scanned by CamScanner

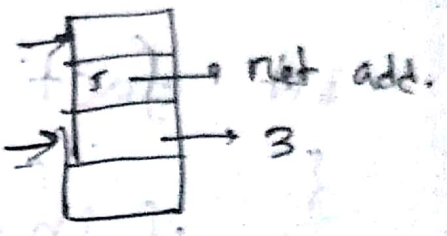
Date : ....../....../......

```
int fact ( int n)
{
    if (n<1) return 1;
    else return n* fact(n-1);
}
```

```
fact:                                   // Parameter পাঠানোর জন্য a0-a3 কে রাখি, এখানে, n a0 তে store করছি।
    addi  $sp, $sp, -8.
    sw    $ra, 4($sp)    // currently return address & a0 এর
    sw    $a0, 0($sp)    // value store করলাম
    slti  $t0, $a0, 1    // slti এর কাজ যদি a0 < 1 হলে t0 এর value হবে 1. নাহলে t0 set হবে 0.
    beq   $t0, $zero, L1 // t0 যদি 0 হয় - branch on equal
                         // t0 যদি 1 হয়, তাহলে
    addi  $v0, $zero, 1
    addi  $sp, $sp, 8    // a0 > 1 হলে L1 এ যাবে।
    jr    $ra.

L1: addi  $a0, $a0, -1   // a0 = a0-1 করলাম
    jal   fact           // symultaneously reg. add কে update করছি, a0 = a0-1 পাঠাচ্ছি
    lw    $a0, 0($sp)    // load করে ফেরত দিলাম।
    lw    $ra, 4($sp)
    addi  $sp, $sp, 8
    mul   $v0, $a0, $v0  // $a0 তে 5 দিয়ে আসছে, 5× f(4) করলাম
    jr    $ra.
```

(margin note beside slti: set ~~shift~~ less than immediate)

n = 3.



slti: --

$a_0 = 3, < 1$? no. $t_0 = 0$

so, beq: jump L1

L1:

$a_0 = a_0 - 1$

$2 4 - 1 = 3.$

will again go fact.