Embedded System Design

MODULE - 1 INTRODUCTION

Embedded System Overview

- Computing Systems are Everywhere. Most of us think of desktop computers like PC's, Laptops, Mainframes, Servers. But there's another type of computing system called as Embedded Computing System.
- Embedded computing systems are computing systems embedded within electronic devices.
- Any computing system other than a desktop computer.
- Embedded systems are found in a variety of common electronic devices such as

1) Consumer Electronics
   - Cell phones
   - Pagers
   - Digital cameras
   - Camcorders
   - Videocassette
   - Recorders
   - Portable video games
   - Calculators
   - Personal digital assistants

2) Home Appliances
   - Microwave Ovens
   - Answering machines
   - Thermostats
   - Home Security System
   - Washing machines
   - Lighting Systems

3) Office Automation
   - Fax machines
   - Copiers
   - Printers
   - Scanners

4) Business Equipment
   - Cash Registers
   - Curbside check-in
   - Alarm Systems
   - Card Readers
   - Product Scanners
   - Automated teller machine

5). Automobiles
- Transmission control
- Cruise control
- Fuel injection
- Antilock brakes
- Active suspension.

- The list of Embedded System goes on.
- By seeing this we can say that any device that runs on electricity either already has or will soon have a computing system embedded within it.
- Embedded computers cost far less than desktop computers, their quantities are huge.

Example: In 1999 a typical American household may have had 1 desktop computer but each one had but 35-50 embedded computers, it is expected to rise to nearly 300 by 2004.
- There was Annual cost growth rate of 17% from several hundred dollars.
- Billion Embedded microprocessor units were sold annually than Hundred million desktop microprocessor units.

Additional Information

- 1st Embedded System was made in MIT instrumentation library, USA. Many types of gates were used for only small applications.

Addition Information continued

- Any device that includes a computer but is not itself a general-purpose computer
- It has H/w as well as S/w
- Embedded system is expected to Respond, monitor, control external environment using sensors & actuators.
- Basically we are talking about Embedding a computer into an appliance & that computer is not expected to use general purpose computer.
- It is part of some larger systems & expected to function without human intervention.

Characteristics of Embedded Systems

Let us see few common characteristics of Embedded System that can be distinguished from other Computing Systems.

1) Single-functioned
2) Tightly contrained
3) Reactive & real time

1) Single functioned
- An embedded system usually executes a specific program repeatedly.
- Example: A pager is always a pager.
- An desktop system executes a variety of programs like spreadsheets, word processors, video games with new programs added frequently.
- There are exceptions.
- i.e. One case is where an embedded system's program is updated with a newer program version.
Example: Some cell phones can be updated in such manner.
Case Two is where several programs are swapped in & out of a system due to size limitations.
Example: Some missiles run one program while in cruise mode, then load a 2nd program for locking onto a target.
We can see that even these exceptions represent systems with a specific function.

2) Tightly Constrained:
- All computing systems have constraints on design metrics, but those on embedded system can be especially tight.
- A design metric is a measure of an implementation's features such as cost, size, performance & power.

- Embedded systems often must cost just a few dollars, must be sized to fit on a single chip, must perform fast enough to process data in real time & must consume minimum power to extend battery life or prevent the necessity of a cooling fan.

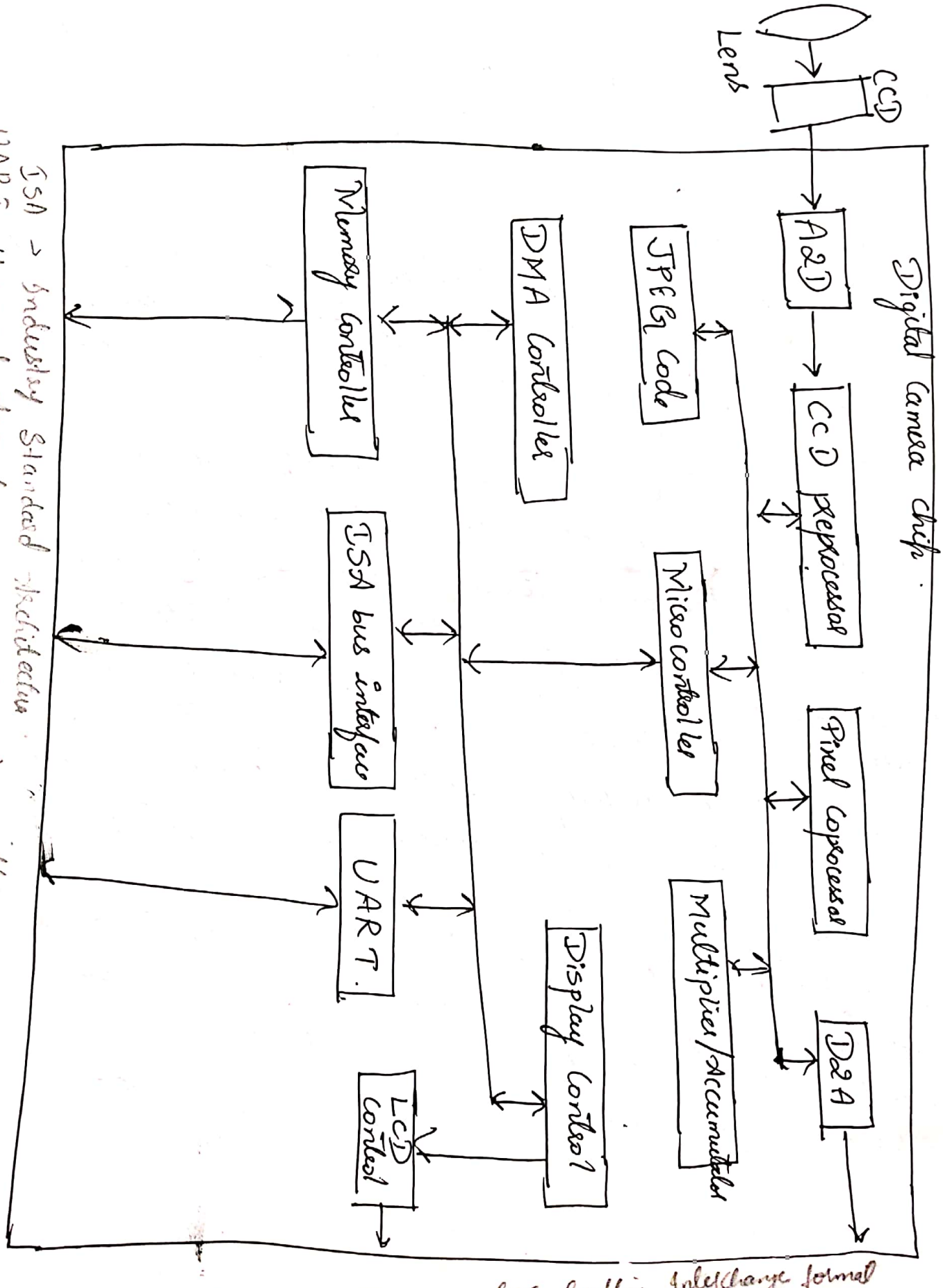
3). Reactive & real time:

- Many embedded systems must continually react to changes in the system's environment & must compute certain result in real time without delay.

Example: A car's cruise controller continually monitors & reacts to speed & brake sensors. - It must compute acceleration or deceleration amounts repeatedly within a limited time, a delayed computation could result in a failure to maintain control of the car.

- In contrast, a desktop system typically focuses on computations with relatively infrequent reactions to i/p devices.
- In addition, a delay in those computations, while perhaps inconvenient to the computer user, typically does not result in a system failure.

Ex:- Embedded System ~~for~~ Example for the characteristics of E.S. (embedded system).



Example

- Consider the digital camera chip shown in fig.
- The charge-coupled device (CCD) contains an array of light-sensitive photocells that capture an image.
- The A2D & D2A circuits contains an array of light-sensitive photocells that capture an ~~image~~.
- The A2D & D2A circuits convert analog images to digital & digital to analog respectively.
- The CCD preprocessor provides commands to the CCD to read the image.
- The JPEG codec compresses & decompresses an image using the JPEG compression standard, enabling compact storage of images in the limited memory of the camera.
- The pixel co-processor aids in rapidly displaying images.
- The memory controller controls access to a memory chip also found in the camera, while the DMA controller enables direct memory access by other devices while microcontroller is performing other functions.
- The UART enables communication with a PC's serial port for uploading video frames, while the ISA bus interface enables a faster connection with a PC's ISA bus.

- The LCD control & display control circuits control the display of images on the camera's liquid-crystal display device.
- The Multiplier / Accumulator circuit performs a particular frequently executed multiply/accumulate computation faster than the microcontroller could.
- At the heart of the system is the Microcontroller which is programmable processor that controls the activities of all the other circuits.
- We can think of each device as a processor designed for a particular task, while the microcontroller is a more general processor designed for general tasks.
- The above example illustrates some of the embedded system characteristics described earlier.
- First, it performs a single function repeatedly. The system always acts as a digital camera, wherein it captures, compresses & stores frames, decompresses & display frames & upload frames.
- Second, it is tightly constrained. The system must be low cost since consumers must be able to afford such a camera. It must be small so that it fits within a standard-sized camera. It must be fast so that it can process numerous images in milliseconds.

It must consume little power so that the camera's battery will last a long time.

Third: However, this particular system does not possess a high degree of the characteristic of being reactive & real time, as it responds only to the pressing of buttons by a user, which, even in the case of an avid photographer, is still quite slow with respect to processor speeds.

1.2 Design Challenge - Optimizing Design Metrics

- The embedded system designer must of course construct an implementation that fulfills desired functionality, but a difficult challenge is to construct an implementation that simultaneously optimizes numerous design metrics

Common Design metrics

- An implementation consists either of a microprocessor with an accompanying program, a connection of digital gates, or some combination.
- A design metric is a measurable feature of a system's implementation.

- Commonly used metrics include.

1) NRE cost (nonrecurring engineering cost):
- The one-time monetary cost of designing the System.
- Once the System is designed, any number of units can be manufactured without incurring any additional design cost, hence the term nonrecurring.

2) Unit cost:
- The monetary cost of manufacturing each copy of the System, excluding NRE cost.

3) Size:
- The physical space required by the System, often measured in bytes for S/w, & gates or transistors for H/w.

4) Performance:
- The execution time of the System.

5) Power:
- The amount of power consumed by the System, which may determine the lifetime of a battery, or the cooling requirements of the IC, since more power means more heat.

6) Flexibility:
- The ability to change the functionality of the System without incurring heavy NRE cost. S/w is typically considered very flexible.

7). Time-to-prototype:
- The time needed to build a working version of the System, which may be bigger or more expensive than the final system implementation, but it can be used to verify the System's usefulness & correctness & to refine the System's functionality.

8). Time-to-market:
- The time required to develop a System to the point that it can be released & sold to to customers.
- The main contributors are design time, manufacturing time & testing time.

9). Maintainability:
- The ability to modify the System after its initial release, especially by designers who did not originally design the System.

10). Correctness:
- Our confidence that we have implemented the System's functionality correctly.
- We can check the functionality throughout the process of designing the System, & we can insert test circuitry to check that manufacturing was correct.

11). Safety: The probability that the System will not cause harm.

- Metrics typically compete with one another
- Improving one often leads to worsening of another.

Example: If we reduce an implementation's size, the implementation's performance may suffer.

- Some observers have compared this phenomenon to a wheel with numerous pins, as illustrated in fig below.

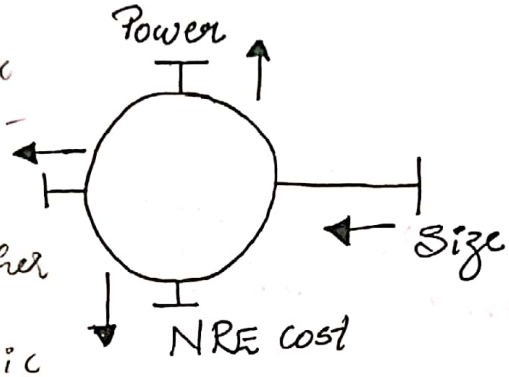


Fig: Design metric competition -
improving one may worsen other

fig: pictorial metric

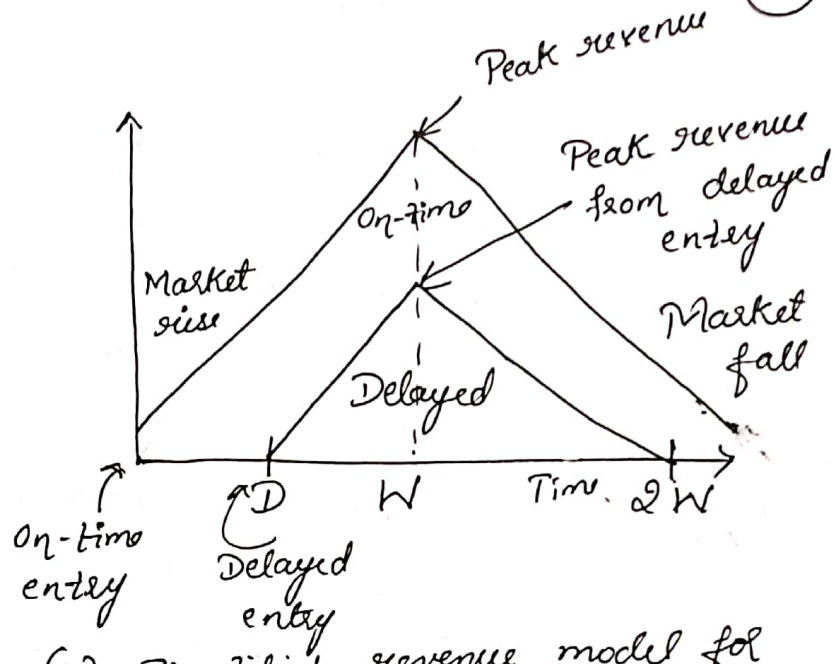
- If you push one pin in, such as size, then the other pins pop out.
- To best meet this optimization challenge, the designer must be comfortable with a variety of H/w & S/w implementation technologies, & must be able to migrate from one technology to another, in order to find the best implementation for a given application & constraints.
- Thus, a designer cannot simply be a H/w expert or a S/w expert, as is commonly the case today.
- The designer must have expertise in both areas.

The Time-to-Market Design Metric

- The Time-to-market is demanding in recent years.
- Introducing an embedded system to the marketplace early can make a big difference in the system's profitability, since market windows for products are becoming quite short, with such windows often measured in months.

Example:

(a) Market window



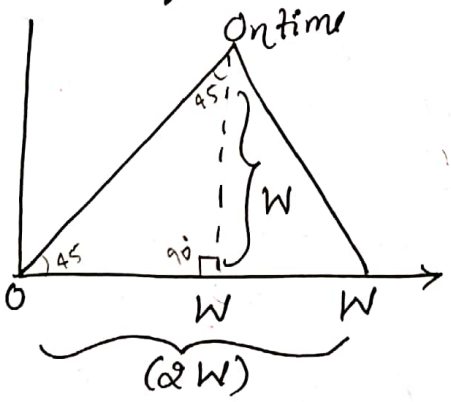
(b) Simplified revenue model for computing revenue loss from delayed entry.

- Fig (a) shows a sample market window during which time a product would have highest sales.
- Missing this window, which means that the product begins being sold further to the right on the time scale, can mean significant loss in sales.
- In some cases, each day that a product is delayed from introduction to the market can translate to a one-million-dollar loss.
- The average time-to-market constraint has been reported as having shrunk to only 8 months!
- The time-to-market constraint is the fact that embedded system complexities are growing due to increasing IC capacities.
- Rapid growth in IC capacity translates into pressure on designers to add more functionality to a system.
- Designers today are being asked to do more in less time.

- Fig (b): To investigate the loss of revenue that can occur due to delayed entry of a product in the market. Assume market rise angle is 45°.
- This model assumes the peak of the market occurs at the halfway point, denoted as 'W' of the product life, & that the peak is the same even for a delayed entry.
- The revenue for an on-time market entry is the area of the triangle labeled On-time, & the revenue for a delayed entry product is the area of the triangle labeled Delayed.
- The revenue loss for a delayed entry is just the difference of these two triangles areas.

$$\% \text{ percentage of revenue loss} = \frac{\text{On-time} - \text{Delayed}}{\text{On-time}} \times 100\%$$

∴ Area of on-time triangle $= \frac{1}{2} \times \text{base} \times \text{height}$



$\tan 45 = 1 = \frac{\text{opp}}{\text{adj}}$

$1 = \frac{\text{opp}}{\text{adj}}$

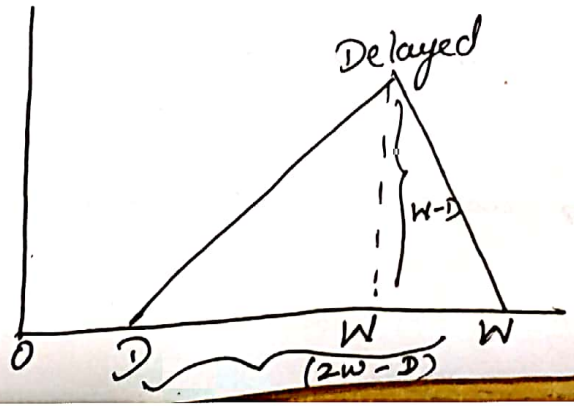
$\text{opp} = \text{adj}$

$= \frac{1}{2} \times 2W \times W$

$= \frac{1}{2} \times 2W^2$

$= W^2.$

∴ Area of delayed triangle $= \frac{1}{2} \times \text{base} \times \text{height}$



$= \frac{1}{2} \times (2W - D) \times (W - D)$

$= \frac{1}{2} [2W^2 - 2WD - WD + D^2]$

$= \frac{1}{2} [2W^2 - 3WD + D^2]$

$= \frac{2W^2}{2} - \frac{3WD + D^2}{2}$

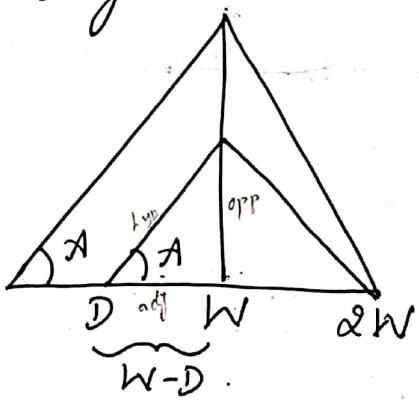
$= W^2 - \frac{D(3W - D)}{2}$

∴ Percentage of Revenue loss $= \frac{W^2 - \left[W^2 - \frac{D(3W-D)}{2}\right]}{W^2} \times 100\%$

$= \frac{W^2 - W^2 + \frac{D(3W-D)}{2}}{W^2} \times 100\%$

$$\boxed{\% \text{ of Revenue loss} = \frac{D(3W-D)}{2W^2} \times 100\%} \Rightarrow \frac{3WD-D^2}{2W^2} \times 100\%$$

2) Derive an equation for percentage revenue loss for any rise angle



$\sin\theta = \frac{opp}{hyp}$

$\cos\theta = \frac{adj}{hyp}$

$\tan\theta = \frac{opp}{adj}$

$\Rightarrow$ W.K.T $\tan A = \frac{Opp}{adj}$

$Opp = \tan A \times adj$

$$\boxed{\text{Revenue loss} = \frac{\text{On-time} - \text{delayed}}{\text{On-time}} \times 100\%} \rightarrow (1)$$

Area of On-time $= \frac{1}{2} \times \text{base} \times \text{height} \rightarrow opp$

$= \frac{1}{2} \times 2W \times \tan A \cdot W$

$$\boxed{\text{Area of On-time} = W^2 \tan A} \rightarrow (2)$$

Area of delay = $\frac{1}{2}$ x base x height

$= \frac{1}{2} \times [(W-D)+W] \times \tan A\,(W-D)$

$$\boxed{\text{Area of delay} = \frac{1}{2} \times [2W-D] \times \tan A\,(W-D)} \longrightarrow (3)$$

Substituting eqn (2) & eqn (3) in eqn (1), we get

$$\text{Revenue loss} = \frac{(\tan A \times W^2) - \left[\frac{1}{2} \times (2W-D) \times \tan A \times (W-D)\right]}{\tan A \times W^2} \times 100\%$$

$$= \frac{W^2 - \left[\frac{1}{2}(2W^2 - 2WD - WD + D^2)\right]}{W^2} \times 100\%$$

$$= \frac{\frac{2W^2 - 2W^2 + 2WD + WD - D^2}{2}}{W^2} \times 100\%$$

$$= \frac{2W^2 - 2W^2 + 2WD + WD - D^2}{2W^2} \times 100\%$$

$$\boxed{\text{Percentage loss} = \frac{3WD - D^2}{2W^2} \times 100\%}$$

3). Determine the revenue loss, if the products lifetime is 52 weeks & the delay in the market is 4 weeks.

⇒ Given,

Life time = 52 Weeks

$W = \frac{52}{2} = 26$ Weeks

Delay $D = 4$ Weeks

% Revenue loss $= \frac{D(3W-D)}{2W^2} \times 100\%$.

$$= \frac{4(3\times 26-4)}{2\times(26)^2}\times 100\%$$

$$= \frac{296}{1352}\times 100\%$$

% Revenue loss = 21.89 %.

4). Determine the revenue loss if the products lifetime is 52 weeks & the delay in the market is 10 weeks

⇒ Given,

$W = \frac{52 \text{ Weeks}}{2}$, $W = 26$ Weeks, $D = 10$ Weeks

$$\% \text{ Revenue loss} = \frac{D(3W-D)}{2W^2}\times 100\%$$

$$= \frac{10(3\times 26-10)}{2\times(26)^2}\times 100\%$$

$$= \frac{680}{1352}\times 100\%$$

% Revenue loss = 50.29 %.

5). Determine the revenue loss, if the product's lifetime is 74 Weeks & the delay in the market is 6 weeks. Derive the formula used for calculation.

⇒ Given,

$W = \frac{74 \text{ Weeks}}{2} = 37$ Weeks, $D = 6$ Weeks

$$\% \text{ Revenue loss} = \frac{D(3W-D)}{2W^2}\times 100\%$$

$$= \frac{6[(3\times 37)-6]}{2\times(37)^2}\times 100\%$$

$$= \frac{630}{2738} \times 100\%$$

% Revenue loss = 23.009 %

6). Derive the formula to measure the percentage revenue loss due to delayed entry of the product to the market. Determine the % revenue loss of the product whose lifetime = 52 Weeks, delayed entry = 4 Weeks.

⇒ Given: $W = \frac{52 \text{ Weeks}}{2} = 26$ Weeks & $D = 4$ Weeks.

$$\% \text{ Revenue loss} = \frac{D[3W - D]}{2W^2} \times 100\%$$

$$= \frac{4[(3 \times 26) - 4]}{2 \times (26)^2} \times 100\%$$

$$= \frac{296}{1352} \times 100\%$$

% Revenue loss = 21.89 %

7). Derive the eqⁿ for % revenue loss for any market rise angle. A product was delayed by 4 weeks in releasing to market. The peak revenue for the product for on-time entry to market would occur after 20 weeks for a market rise angle of 45°. Determine the % revenue loss.

⇒ The peak revenue for the product for On-time entry to market would occur is $W' = 20$ Weeks.

$D = 4$ Weeks.

$$\% \text{ Revenue loss} = \frac{D[3D - D]}{2W^2} \times 100\%$$

$$= \frac{4[3 \times 20) - 4]}{2 \times (20)^2} \times 100\%$$

$$= \frac{224}{800} \times 100\%$$

% Revenue loss = 28%

8). The life time of a product is 64 Weeks. If the product is delayed by 6 Weeks to market, What is the % of revenue loss?

$\Rightarrow$ $W = \frac{64 \text{ Weeks}}{2} = 32$ Weeks, $D = 6$ Weeks

$$\% \text{ Revenue loss} = \frac{D[3W - D]}{2W^2} \times 100\%$$

$$= \frac{6[(3 \times 32) - 6]}{2 \times (32)^2} \times 100\%$$

$$= \frac{540}{2048} \times 100\%$$

% Revenue loss = 26.36%

9). Using the revenue model, compute the % revenue loss if $D = 5$ & $W = 10$. If the company whose product entered the market on-time earned a total revenue of $25 million, How much revenue did the company that entered the market 5 Weeks late loss?

$\Rightarrow$ $$\% \text{ Revenue loss} = \frac{D[3W - D]}{2W^2} \times 100\%$$

$$= \frac{5[(3 \times 10) - 5]}{2 \times (5)^2} \times 100\%$$

% Revenue loss = 62.5%

Revenue loss = $25,000,000 x 0.625

Revenue loss = $ 15,625,000

NRE & Unit Cost Design Metrics

NRE cost : NRE cost is the nonrecurring engineering cost. It is the one time monetary cost of designing the system.

Total cost:

Total Cost = NRE Cost + Unit cost x Number of Units

Per-product cost:

$$\text{Per-product cost} = \frac{\text{Total cost}}{\text{No of Units}}$$

$$= \frac{\text{NRE cost} + \text{unit cost} \times \text{No of Units}}{\text{No of Units}}$$

$$= \frac{\text{NRE cost}}{\text{No. of Units}} + \frac{\text{Units cost} \times \text{No of units}}{\text{No of Units}}$$

$$\textbf{Per-product Cost} = \frac{\text{NRE Cost}}{\text{No. of Units}} + \text{Unit Cost}$$

FORMULA

$$\text{Added cost} = \frac{\text{NRE}}{\#\text{ Units produced.}}$$

Example:

Technology A would result in a NRE cost of $2,000 & Unit cost of $100, Technology B would have an NRE cost of $30,000 & Unit cost of $30, & Technology C would have an NRE cost of $100,000 & unit cost of $2
i) Plot total cost versus the number of units produced.
ii) Plot per-product cost versus the number of units produced.

⇒ i) Total cost = NRE cost + [Unit cost × No of Units]

Technology A
NRE cost = $2000
Unit cost = $100

No of Units	Total Cost in $
0	2000
1	2100
100	12,000
200	22,000
400	42,000
800	82,000
1200	1,22,000
1600	1,62,000
2000	2,02,000
2400	2,42,000

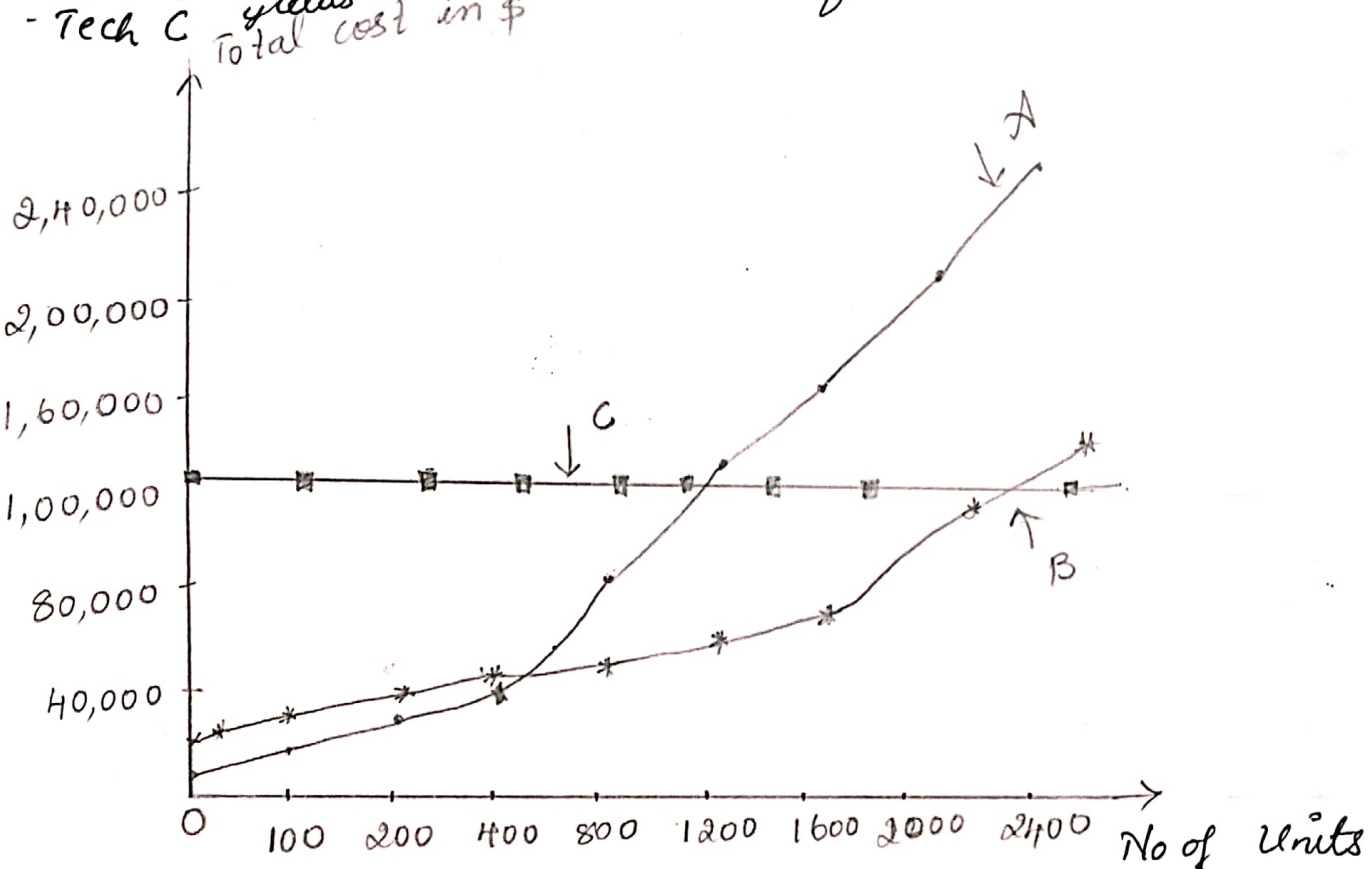
Technology B
NRE cost = $ 30,000
Unit cost = $30

No of Units	Total Cost in $
0	30,000
1	30,030
100	33,000
200	36,000
400	42,000
800	54,000
1200	66,000
1600	78,000
2000	90,000
2400	1,02,000

Technology C
NRE cost = $ 100,000
Unit cost = $ 2

No of Units	Total cost in $
0	1,00,000
1	1,00,002
100	1,00,200
200	1,00,400
400	1,00,800
800	1,01,60
1200	1,02,400
1600	1,03,200
2000	1,04,000
2400	1,04,800

- From the plot of 3 technologies, Tech A yields lowest total cost for low volumes, namely for volumes betn 1 & 400
- Tech B yields lowest total cost for volumes betn 400$ & 2500.
- Tech C yields lowest cost for volumes above 2500



ii). Per product cost = $\frac{\text{NRE cost}}{\text{No of Units}}$ + Unit cost

OR

Per product cost = $\frac{\text{Total cost}}{\text{No of Units}}$.

Technology A
NRE cost = $2000
Unit cost = $100

No of Units	per product cost in $
1	2,100
100	120
400	105
800	102.5
1200	101.66
1600	101.25

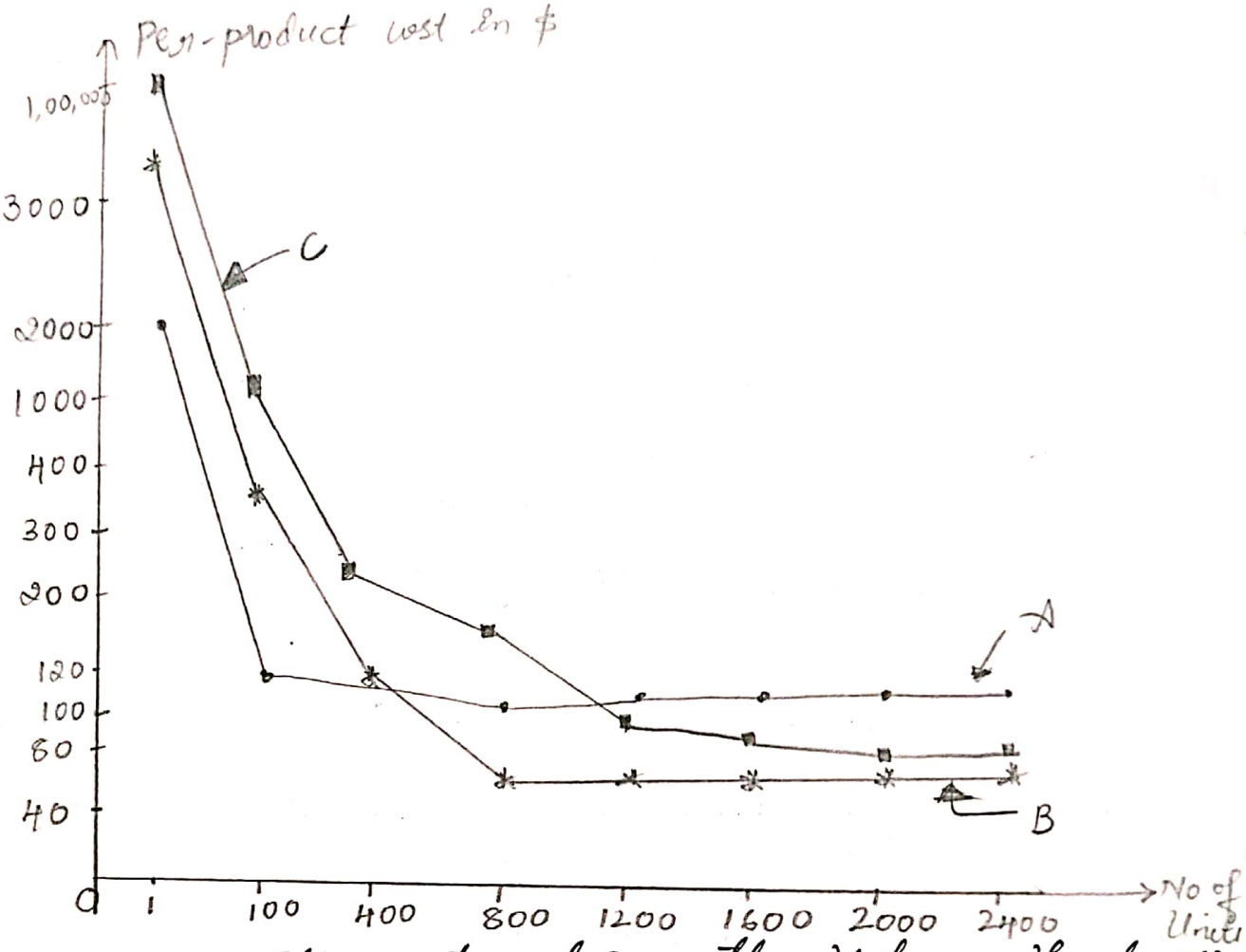
Technology B
NRE cost = $39,000
Unit cost = $30

No of Units	per product in $
1	30,030
100	330
400	105
800	67.5
1200	55
1600	48.75

Technology C
NRE cost = $1,09,00
Unit Cost = $2

No of Units	per-produ in $
1	1,00,002
100	1002
400	252
800	127
1200	85.33
1600	64.5

Tech A		Tech B		Tech C	
No of units	per product in $	No of Units	per product in $	No of Units	per product in $
2000	101	2000	45	2000	52
2400	100.83	2400	42.5	2400	43.66



- Above fig illustrates larger the Volume, the lower the per product cost, since the NRE cost can be distributed over more products.
- Clearly one must consider the revenue impact of both time-to-market & per product cost, as well as all the other relevant design metrics when evaluating different technologies.

The Performance Design Metric

- Performance of a system is a measure of how long the system takes to execute our desired tasks.
- Performance is widely used design metric in marketing an embedded system & also one of the most abused.

- Many metrics are commonly used in reporting System performance. Such as clock frequency or instructions per second.
- Important is how long the system takes to execute our application.

-Example: In terms of performance, we care about how long a digital camera takes to process an image.
- The camera's clock frequency or instructions per second are not the key issues -
- One camera may actually process images faster but have a lower clock frequency than another camera.
- Suppose we have a single task that will be repeated over & over, such as processing an image in a digital camera.
- The 2 main measures of performance are

Latency or Response time:
- The time bet" the start of the task's execution & the end.

Example: Processing an image may take 0.25 sec.

Throughput:
- The number of tasks that can be processed per unit time.

Example: A camera may be able to process 4 images per second.

- Throughput is not always just the number of tasks times latency.
- A system may be able to do better than this by using parallelism, either by starting one task before finishing the next one or by processing each task concurrently.

Example: A digital camera might be able to capture & compress the next image, while still storing the previous image to memory.

- In embedded systems, performance at a very detailed level is also often of concern.
- In particular, a signal changes may have to be generated or measured within some number of nanoseconds.

Speed up:

- It is a common method of comparing the performance of 2 systems. The speedup of system A over system B is determined as

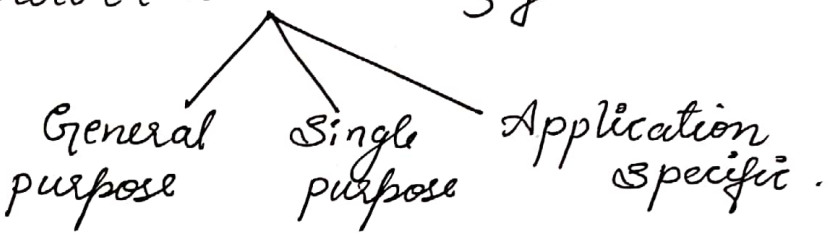
$$\text{Speedup of A over B} = \frac{\text{performance of A}}{\text{performance of B}}.$$

- Performance could be measured either as latency or as throughput depending of interest.
- Suppose the speedup of camera A over camera B is 2.
- Then we also can say that A is 2 times faster than B & B is 2 times slower than A.

Processor Technology

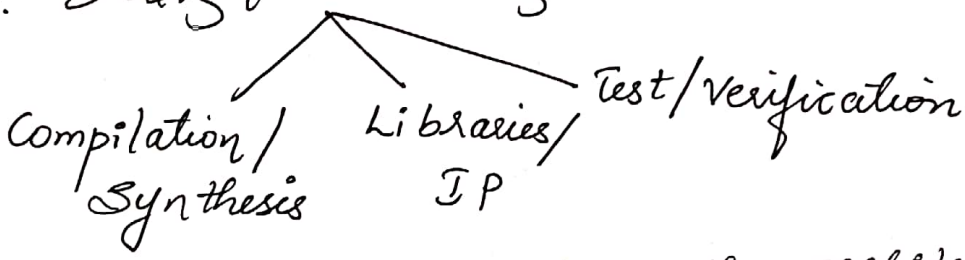
- Technology is defined as accomplishing a task, especially using technical processes, methods, or knowledge.
- We consider 3 types of technologies

1) Processor Technology



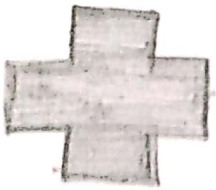
2) IC Technology

3) Design Technologies



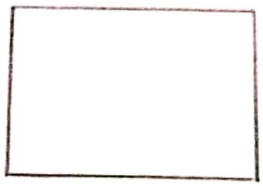
- Processor technology relates to the architecture of the computation engine used to implement a systems desired functionality.
- The term processor is usually associated with programmable s/w processors.
- There are many other, nonprogrammable, digital systems as being processors.
- Such processors differs in its specialization towards a particular function. Example: Image compression

- We illustrate this concept graphically in fig below.



(a)

```
total = 0
for i = 1 to N
loop
  total += M[i]
end loop
```



(b)

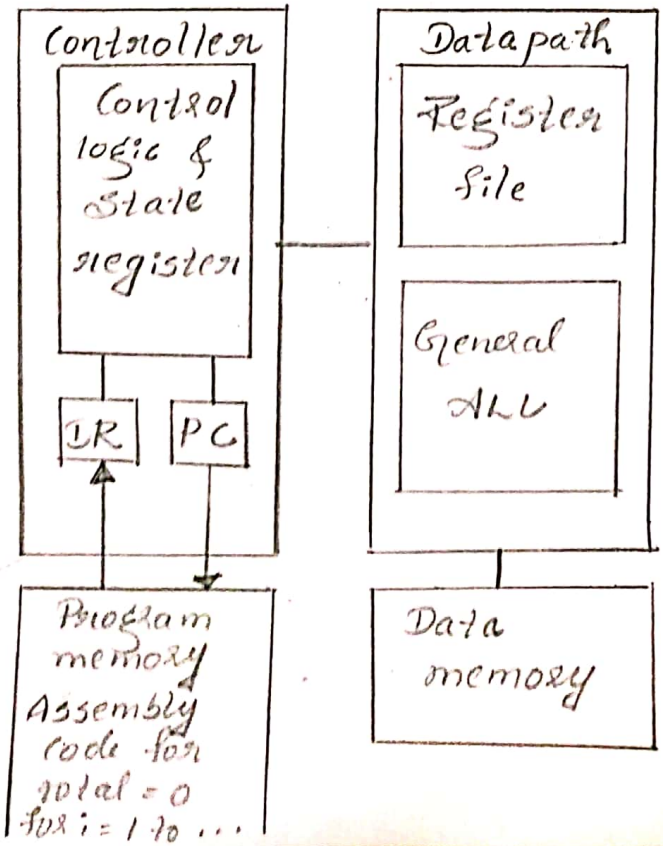
(c)

(d)

fig: Processors vary in their customization for the problem at hand. (a) desired functionality. (b) General-purpose processor. (c) Application-specific processor. (d) Single-purpose processor.

- Three main processor technologies are.
  1). General-purpose processors (S/w)
  2). Single-purpose processors (H/w)
  3). Application specific processors (ASIP's).

- General-Purpose Processors (S/w).



- The design of a general purpose processor or microprocessor builds a programmable device that is suitable for a variety of applications to maximize the number of devices sold.
- The features of such processors are
  1) Program Memory - The designer of such a processor does not know what program will run on the processor, so the program cannot be built into the digital circuit.
  2) Data path - The data path must be general enough to handle a variety of computations, so such a datapath typically has a large register file & one or more general-purpose arithmetic-logic units (ALU's).
- An Embedded system designer, however, need not be concerned about the design of a general-purpose processor.
- An Embedded system designer simply uses a general-purpose processor, by programming the processor's memory to carry out the required functionality.
- Fig shows the simple architecture of a general-purpose processor implementing the array-summing functionality.
- The functionality is sorted in the program memory.
- The controller fetches the current instruction as indicated by the program counter (PC) into the instruction register (IR).
- It then configures the datapath for this instruction & executes the instruction.
- It then determines the next instruction address, sets the PC to this address & fetches again.

Advantages

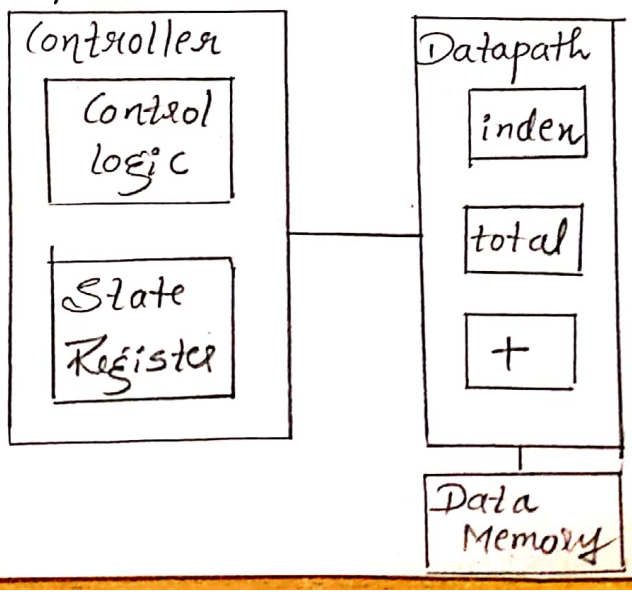
1. Time-to-market & NRE cost are low because the designer must only write a program but not do any digital design.
2. Flexibility is high because changing functionality requires changing only the program
3. Unit cost may be low in small quantities compared with designing our own processors.
4. Performance may be fast for computation intensive applications.

Design metric Drawbacks

1. Unit cost may be relatively high for large quantities
2. Performance may be slow for certain applications
3. Size & power may be large due to unnecessary processor H/w.

Single Purpose Processor

- A single purpose processor is a digital circuit designed to execute exactly one program. It is also known as co-processor, accelerator or peripheral.



- The above fig shows the architecture of a single-purpose processor.
- It contains only the components needed to execute a single program.
- It has no program memory
- The datapath contains only the essential components for this program i.e, 2 registers & one adder.
- Since the processor only executes this one program we hardwire the programs instructions directly into the control logic & use a state register to step through those instructions, so no program memory is necessary.

Design metric advantages or Benefits

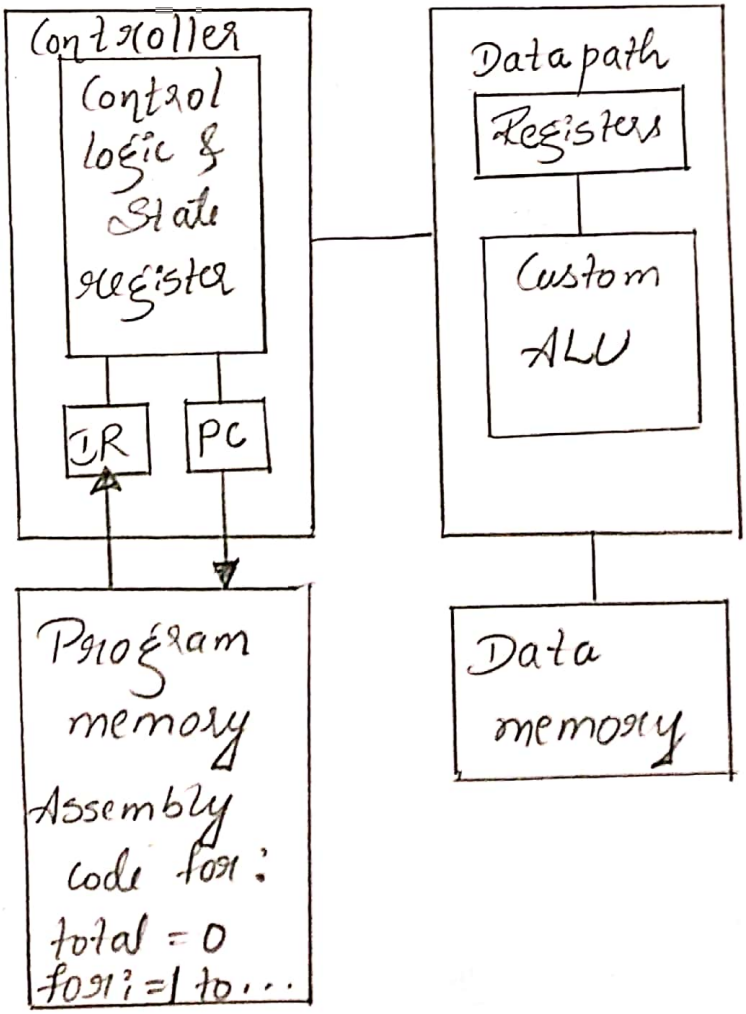
- Size & power may be small.
- Performance may be fast.
- Unit cost may be low for large quantities.

Design metric disadvantages

- Design-time & NRE cost may be high.
- Flexibility is low.
- Unit cost high for small quantities.
- Performance may not match general-purpose processor for some applications.

Application Specific Processors (ASIPs)

- An application specific instruction set processor (ASIP) can serve as a compromise bet" the general purpose processor & single purpose processors.
- An ASIP is a programmable processor optimized for a particular class of applications having common characteristics such as embedded control, digital-signal processing or telecommunications.
- Designer of such a processor can optimize the datapath for the application class, perhaps adding special functional units for common operations & eliminating other infrequently used units



- The features of ASIP are
- Program memory
- Optimized datapath
- Special functional units.

Advantages:

- Flexibility
- Good performance, power & size.

Disadvantages

- Require large NRE cost to build the processor itself & to build a compiler, if these items dont already exists.

Types of ASIP

- ASIP's are of 2 types
  - Microcontrollers
  - Digital Signal processors.

Microcontroller

- A microcontroller is a microprocessor that has been optimized for embedded control applications. Such applications typically monitors & set numerous single bit control signals but do not perform large amount of data computations.
- Thus microcontrollers tend to have simple datapaths that excel at bit-level operations & at reading & writing external bits.
- Microcontrollers are used in control applications like Serial communication peripherals, timers, counters,

pulse width modulators & analog-digital converters.

Advantages

- Single chip
- Smaller (compact) &
- Low cost.

Digital Signal Processors (DSP's):

- A DSP is a microprocessor, designed to perform common operations on digital signals, which are the digital encoding of analog signals like Video & audio.
- These operations carry out common signal processing tasks like signal filtering, transformations or combination. Such operations are math-intensive including operations like multiply & add or shift & add.
- To support such operations, a DSP may have special purpose datapath components such as multiply-accumulate unit, which can perform computation like $T = T + M[i] * K$ using only one instruction.
- DSP programs often manipulate large arrays of data. A DSP may also include special H/w to fetch sequential data memory locations in Hel with other operations to further speed execution.

MODULE-2

PROCESSORS ARCHITECTURE

- In today's era there are variety of multiprocessor exist.
- In this module we will study about advances in the processor technology, Superscalar & Vector Processors architecture in detail.

Advanced Processor Technology

- Architectural families of modern processors are introduced below with the underlying microelectronics/packaging technologies.
- The coverage spans from VLSI microprocessors used in workstations or multiprocessors to heavy-duty processors used in mainframes & supercomputers.
- Major processor families to be studied include the CISC, RISC, Superscalar, VLIW (Very long Instruction Word), Superpipelined, vector processors.
- Vector & Scalar processors are for numerical computations

Design Space of Processors.

- Various processor families can be mapped onto a co-ordinated space of clock rate versus cycles per instruction (CPI) as given in fig below.
- As implementation technology evolves rapidly, the clock rates of various processors are gradually moving from low to higher speeds toward the right of the design space.
- Another trend is that processor manufacturers are trying to lower the CPI rate using H/w & S/w approaches.

- Based on these trends, the mapping of processors in fig below reflects their implementation during the past decade.
- As time passes, some of the mapped ranges may move toward the lower right corner of the design space.

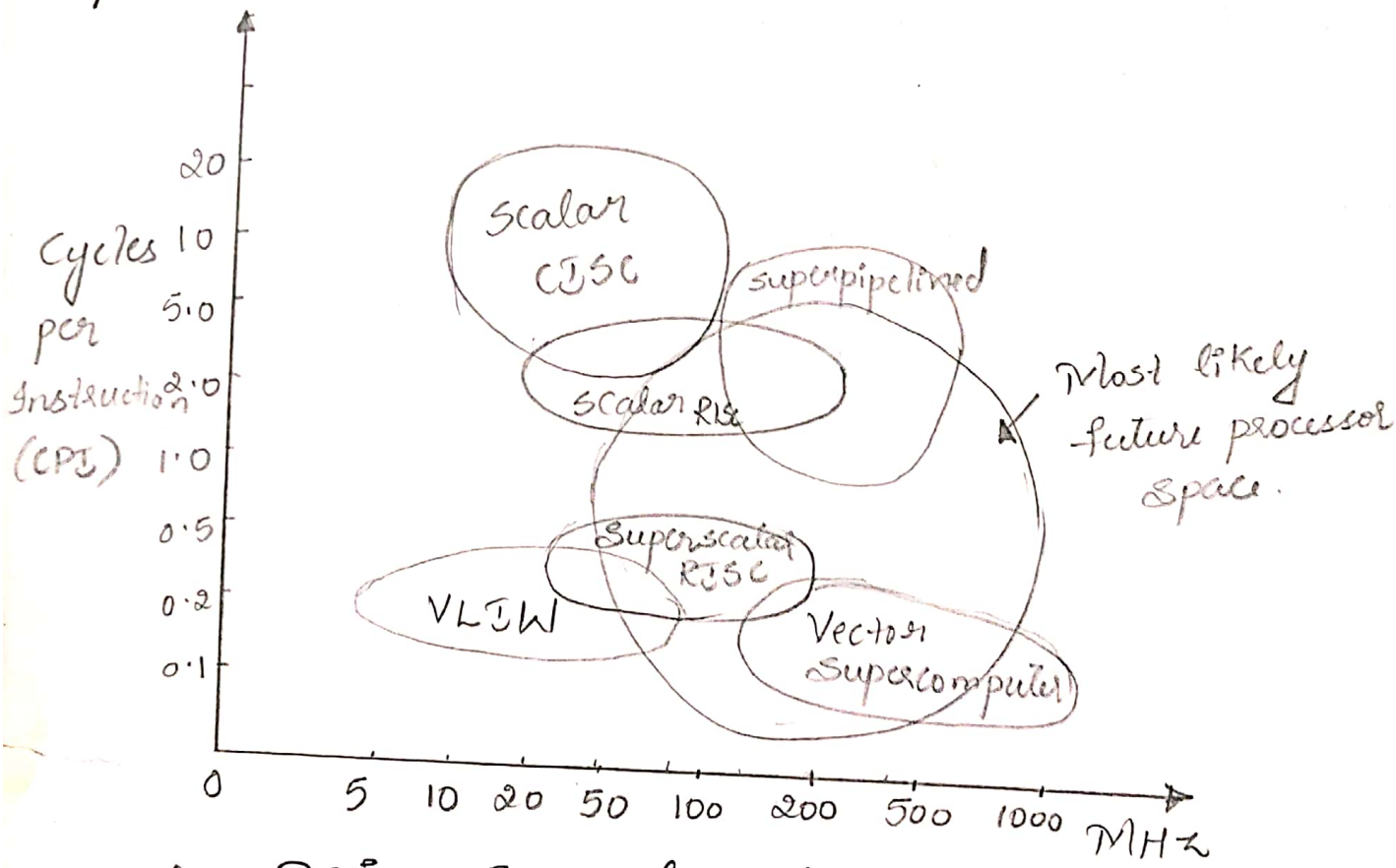


Fig: Design space of modern processor families.

The Design Space

- **Scalar CISC (Complex-instruction-set computing)**
- Conventional processors like Intel i486, M68040, VAX/8600, IBM 390 etc fall into the family known as complex-instruction set computing (CISC) architecture
- The typical clock rate of today's CISC processors ranges from 33 to 50 MHz.
- With microprogrammed control, the CPI of different

CISC instructions varies from 1 to 20.
- ∴ CISC processors are at the upper left of the design space.

• Scalar RISC (Reduced instruction set computing)
- Today's reduced instruction set computing (RISC) processors, such as the Intel i860, SPARC, MIPS R3000, IBM RS/6000 etc have faster clock rates ranging from 20 to 120 MHz.
- With the use of hardwired control, the CPI of most RISC instructions have been reduced to one to two cycles.

• Superscalar RISC
- A special subclass of RISC processors are the superscalar RISC processors which allow multiple instructions to be issued simultaneously during each cycle.
- Thus the effective CPI of a superscalar processor should be lower than that of a generic scalar RISC processor.
- The clock rate of superscalar processors matches that of scalar RISC processors.

• VLIW (Very Long instruction Word)
- VLIW architecture uses even more functional units than that of a superscalar processor.
- The CPI of a VLIW processor can be further lowered.
- Due to the use of very long instructions (256 to 1024 bits per instruction), VLIW processors have been mostly implemented with microprogramme control.

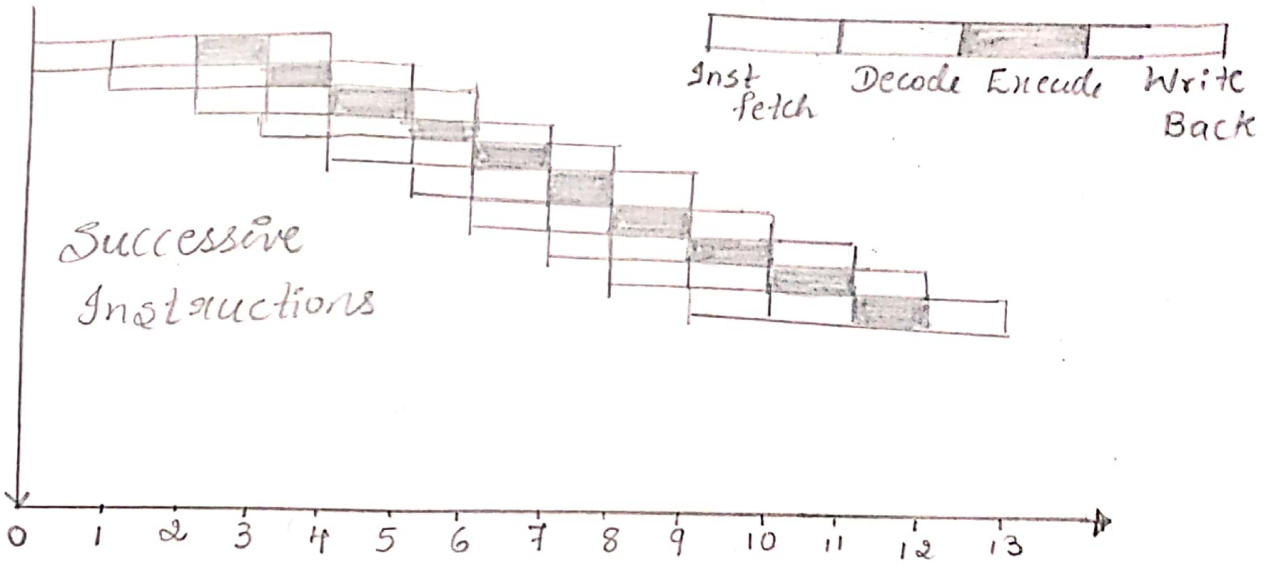
- Thus, the clock rate is slow with the use of read-only memory (ROM).
- A large number of microcode access cycles may be needed for some instructions.

• Superpipelined processors
- It uses multiphase clocks with a much increased clock rate ranging from 100 to 500 MHz.
- CPI rate is rather high unless superpipelining is practiced jointly with multi-instruction issue.
- The processors in vector supercomputers are mostly superpipelined & use multiple functional units for concurrent scalar & vector operations.
- The effective CPI of a processor used in a supercomputer should be very low, positioned at the lower right corner of the design space.
- However, the cost increases appreciably if a processor design is restricted to the lower right corner.

Instruction Pipelines

- The execution cycle of a typical instruction includes four phases
  1) Fetch
  2) Decode
  3) Execute
  4) Write-back.

- These instruction phases are often executed by an instruction pipeline as demonstrated in fig below



(a) Execution in a base Scalar processor

- In other words, we can simply model an Instruction processor by such a pipeline structure.

Example: The pipeline, like an industrial assembly line, receives successive instructions from its i/p end & executes them in a streamlined, overlapped fashion as they flow through.

- A pipeline cycle is defined as the time required for each phase to complete its operation, assuming equal delay in all phases (pipeline stages).

Basic Definitions:

1) Instruction pipeline Cycle:
The clock period of the instruction pipeline

2) Instruction Issue Latency:
The time (in cycles) required bet" the issuing of 2 adjacent instructions.

3). Instruction issue rate:
The number of instructions issued per cycle. also called the degree of a superscalar processor.

4). Simple operation Latency:
- Simple operations make up the vast majority of instructions executed by the machine, such as integer adds, loads, stores, branches, moves etc.
- Complex operations are those requiring an order-of-magnitude longer latency, such as divides, cache misses etc.
- These Latencies are measured in number of cycle

5). Resource Conflicts:
This refers to the situation where 2 or more instructions demand use of the same functional unit at the same time.

- A Base scalar processor is defined as a machine with one instruction issued per cycle, a one-cycle latency for a simple operation, & one-cycle latency bet" instruction issues.
- The instruction pipeline can be fully utilized if successive instructions can enter it continuously at the rate of one per cycle as shown in fig @.

- The instruction issue latency can be more than one cycle for various reasons.

Example :- If the instruction issue latency is two cycles per instruction, the pipeline can be underutilized, as demonstrated in fig (b) below.

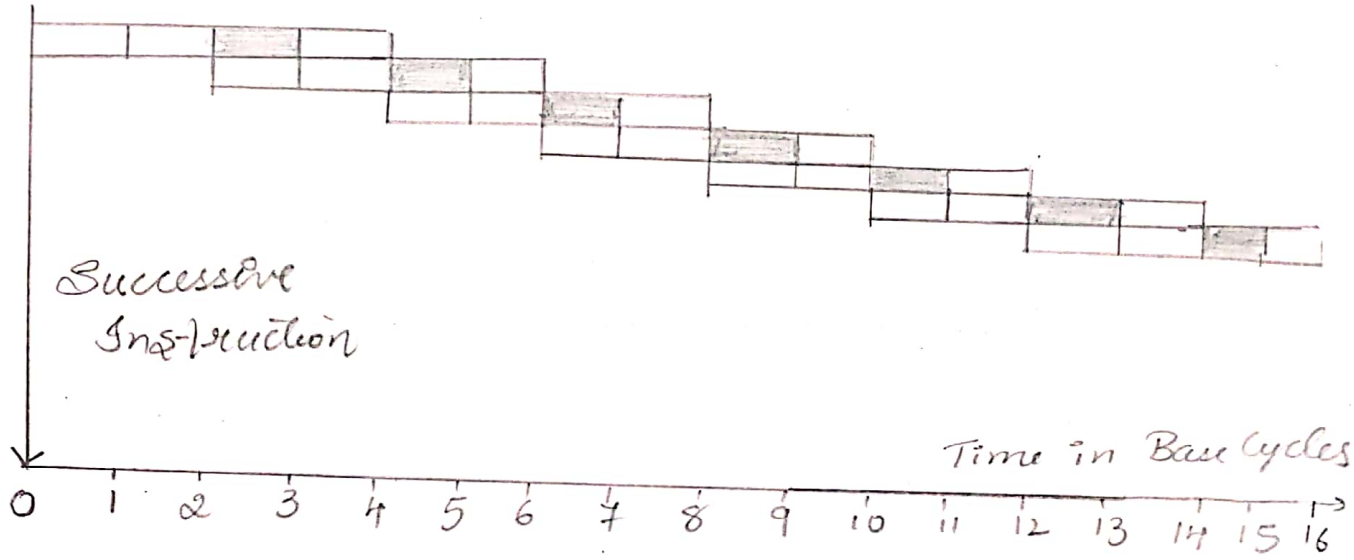


Fig (b) Underpipelined with two cycles per inst issue

- Another underpipelined situation is shown in fig (c) below, in which the pipeline cycle time is doubled by combining pipeline stages.
- In this case, the fetch & decode phases are combined into one pipeline stage, & execute & write-back are combined into another stage.
- This will also result in poor pipeline utilization

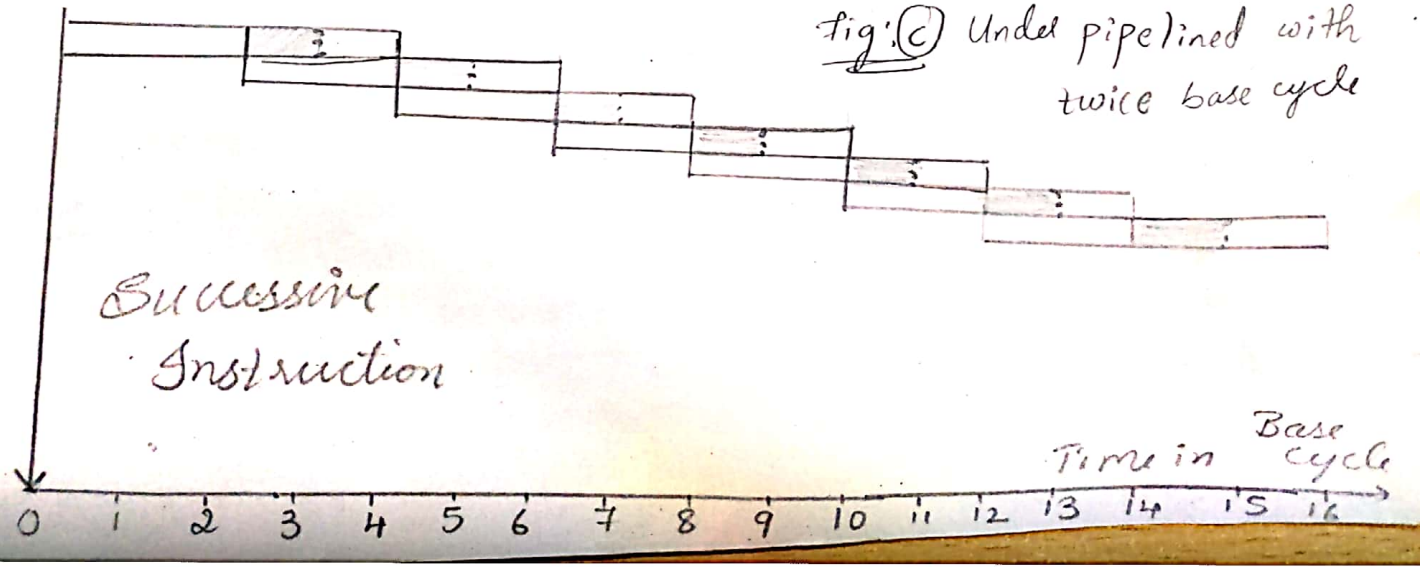
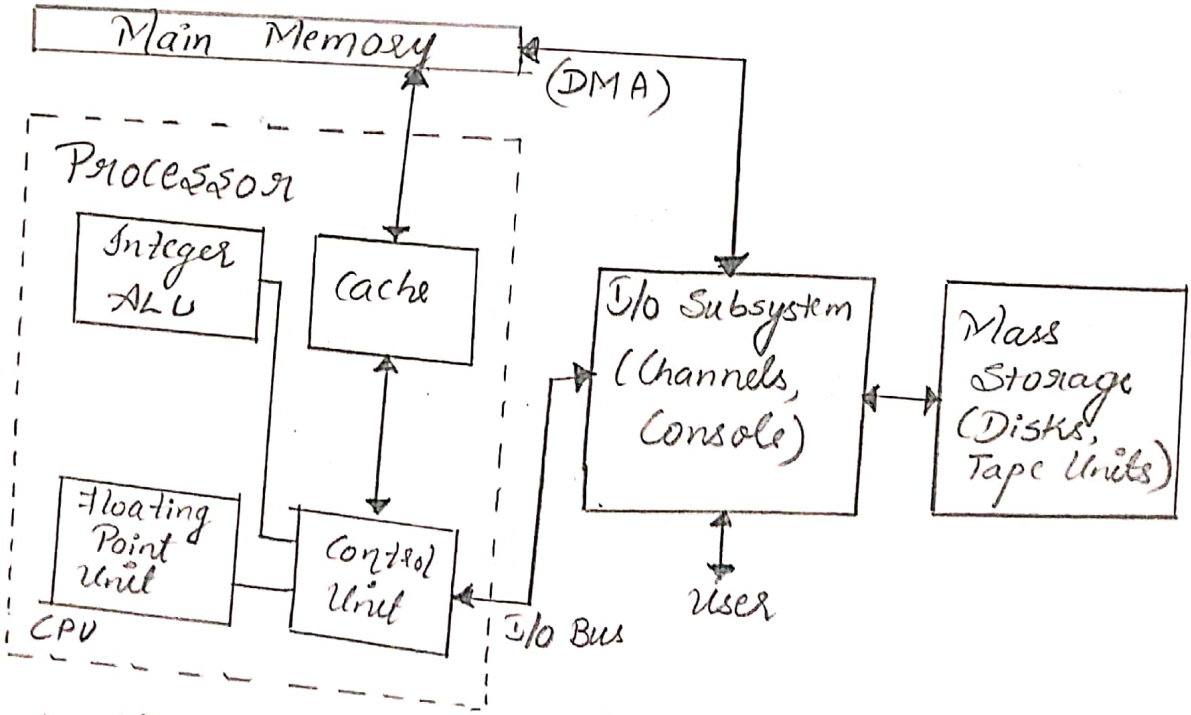


Fig (c) Under pipelined with twice base cycle

- The effective CPI rating is 1 for the ideal pipeline shown in fig ⓐ & 2 for the case in fig ⓑ.
- In fig ⓒ the clock rate of the pipeline has been lowered by one-half.
- According to fig ⓑ & fig ⓒ will reduce the performance by one-half, compared with ideal case fig ⓐ for the base machine.

Processors & Co-processors

- The central processor of a computer is called the Central processing Unit (CPU) as given in fig below



ALU - Arithmetic & Logic Unit
DMA - Direct Memory Access
CPU - Central processing Unit.

fig ⓐ CPU with built-in-floating point Unit.

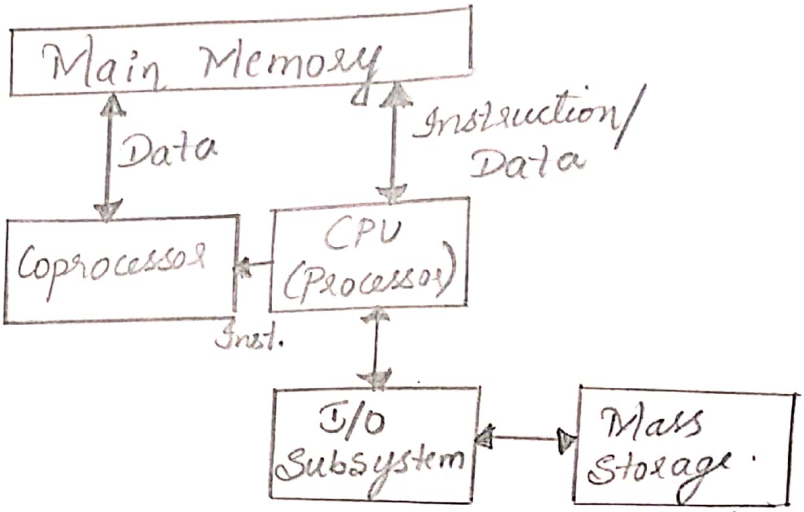


Fig (b) CPU with an attached co-processor
Architectural model of a basic scalar computer system

- This CPU is essentially a scalar processor, which may consist of multiple functional units such as an integer arithmetic & logic unit (ALU), a floating-point accelerator etc.
- The floating point unit can be built on a co-processor (Fig (b)) which is attached to CPU.
- The <u>co-processor</u> executes instruction dispatched from the CPU.
- A co-processor may be a floating point accelerator executing scalar data, vector processor executing vector operands, a DSP or a Lisp processor executing AI (Artificial intelligence) programs.
- Co-processor cannot handle I/O operations.

<u>Co-processor - Processor pairs & Characteristics</u>

- Table below lists some processor - coprocessor pairs developed in recent years to speed up numerical computations.
- Co-processors cannot be used alone.

- The processor & co-processor operate with a host-back-end relationship & compatibility betⁿ them is a necessity.
- Co-processors attached processors are also called as slave processors.
- An attached processor may be more powerful than its host.

Example: Cray Y-MP is a back-end processor driven by a small mini computer.

Coprocessor	Compatible Processor	Coprocessor Characteristics
Intel 8087	Intel 8086/8088	5MHz, 70 cycles for Add & 700 cycles for log.
Intel 80287	Intel 80286	12.5MHz, 30 cycles for Add & 264 cycles for log.
Intel 387DX	Intel 386DX	33MHz, 12 cycles for Add & 210 cycles for log.
Intel i486	Intel i486 (the same chip)	33MHz, 8 cycles for Add & 171 cycles for log
Motorola MC68882	Motorola MC68020/68030	40MHz, 56 cycles for Add & 574 cycles for log.
Weitek 3167	Intel 386DX	33MHz, 6 cycles for Add 365 cycles for log by s/w emulation
Weitek 4167	Intel i486	33MHz, 2 cycles for Add & not available for log

Instruction Set Architecture (lowest level visible to programmer)

- The Instruction Set of a computer specifies the primitive commands or machine instructions that a programmer can use in programming the machine.
- The complexity of an instruction set is attributed to the instruction formats, addressing modes, general-purpose registers, opcode spec specifications & flow control mechanisms used.
- Based on past experience in processor design, two schools of thought on instruction-set architectures have evolved, namely CISC & RISC.

prog not concerned abt gate, delays.

Complex Instruction Set Computer [CISC]

ALU, registers are not inst. The way they are put

- In early days of computer history, most computer families started with an Instruction set which was rather simple.
- The main reason for being simple then was the high cost of H/w.
- The H/w cost has dropped & S/w cost has gone up steadily on in the past 3 decades.
- The Semantic gap betn HLL (High level language) features & computer architecture has widened.
- More & More functions have been built into the H/w making the instruction set very large & very complex.
- The growth of instruction sets was encouraged & user defined instruction sets were implemented using microcodes in some processors for special purpose applications.

- A typical CISC instruction set contains approximately 120 to 350 instructions using variable instruction/data formats, uses small set of 8 to 24 general-purpose registers (GPRs) & executes large number of memory reference operations based on more than a dozen addressing modes.
- Many HLL statements are directly implemented in H/w/firmware in a CISC architecture.
- This may simplify the compiler development, improve execution efficiency, & allow an extension from scalar instructions to vector & symbolic instructions.

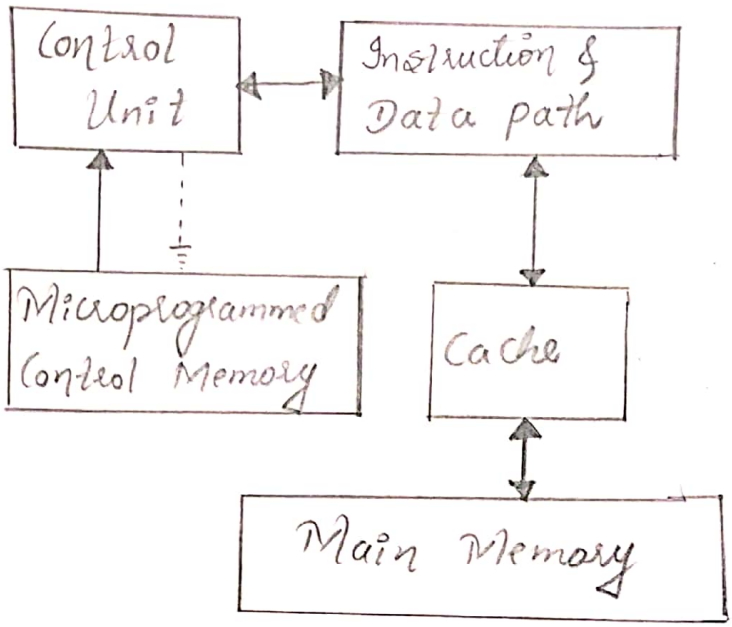
Reduced Instruction Set Computer [RISC].

- Started with RISC & gradually moved to CISC in 1980's.
- After 2 decades of using CISC processors, computer users began to re-evaluate the performance relationship bet^n instruction-set architecture & available H/w/S/w technology.
- Later, Computer scientists realized that only 25% of the instructions of a complex instruction set are frequently used about 95% of time.
- This implies that about 75% of H/w supported instructions often are not used at all.
- Why should we waste valuable chip area for rarely used instruction? was arised.
- With low-frequency elaborate instructions demanding long microcode to execute them, it may be

more advantageous to remove them completely from H/w & rely on S/w to implement them.

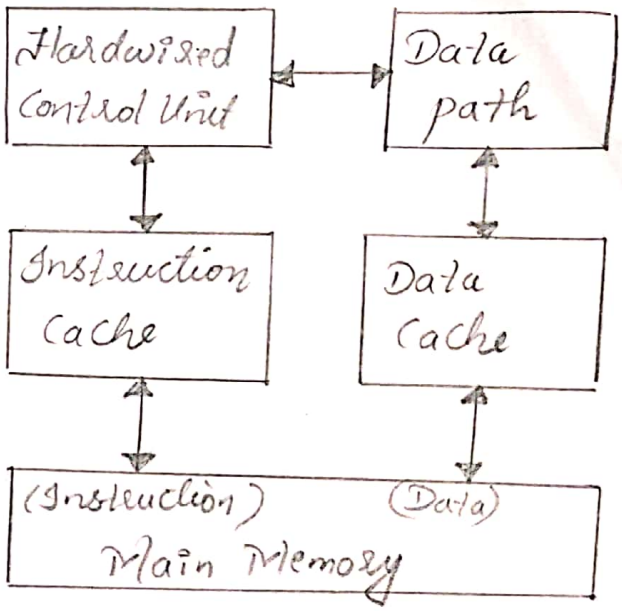
- Even if S/w implementation is slow, the net result will be still a plus due to their low frequency of appearance.
- Pushing rarely used instructions in S/w will vacate chip areas for building more powerful RISC or Superscalar processors, even with on-chip caches or floating-point units.
- A Typical RISC contains less than 100 instructions with a fixed instruction formats (32 bits).
- Only 3 to 5 simple Addressing modes are used.
- Most instructions are register-based.
- Memory access is done by load/store instructions only.
- A large register file (at least 32) is used to improve fast context switching among multiple users & most instructions execute in one cycle with hardwired control.
- Because of the reduced in instruction set complexity, the entire processor is implementable on a single VLSI chip.
- The resulting benefits include a higher clock rate & a lower CPI, which lead to higher MIPS (Microprocessor without interlocked pipeline stages) ratings as reported on commercially available RISC/superscalar processors.

Architectural Distinctions betⁿ CISC & RISC



(a) CISC architecture with microprogrammed control & unified cache

- It uses Unified Cache for holding both instructions & data ∴ They must share same data/instruction path.
- The use of microprogrammed control can be found
- Split caches & hardwired control are used
- Hardwired control seen in CISC which will reduce CPI effectively to 1 instruction per cycle if pipeling is carried out perfectly.



(b) RISC architecture with hardwired control & split instruction cache & data cache

- It uses separate instruction & data caches with different access paths.
- Hardwired control found in RISC.
- Split caches & Hardwired control are not exclusive in RISC.
- CPI < 1.5.

Characteristics of CISC & RISC Architectures

- We compare the main features of RISC & CISC processors which involves 5 area: Instruction sets, addressing modes, register file & cache design, clock rate & expected CPI, & control mechanisms.

Architectural Characteristic	Complex Instruction Set Computer (CISC)	Reduced Instruction Set Computer (RISC).
Instruction-set size & Instruction format	Large set of instructions with variable formats (16-64 bits per instruction).	Small set of instructions with fixed (32-bit) format & most register based instructions.
Addressing modes	12-24	Limited to 3-5
GPR (General purpose Register) & Cache design	8-24 GPRs, mostly with a unified cache for instructions & data, recent designs also use split caches	Large numbers (32-192) of GPRs with mostly split data cache & instruction cache.
Clock rate & CPI	33-50 MHz in 1992 with a CPI bet^n 2 & 15	50-150 MHz in 1993 with one cycle for almost all instructions & an average CPI < 1.5.
CPU Control	Most microcoded using control memory (ROM), but modern CISC also uses hardwired control.	Most hardwired without control memory.

I CISC Scalar Processors

- A scalar processor executes with scalar data
- The simplest scalar processor executes integer instructions using fixed-point operands.
- More capable scalar processors execute both integer & floating-point operations.
- A modern scalar processor may posses both an integer unit & a floating-point unit in the same CPU.
- Based on a complex instruction set, a CISC scalar processor can be built either with a single chip or with multiple chips mounted on a processor board
- In the ideal case, a CISC scalar processor should have a performance equal to that of the base scalar processor

Representative CISC Processors

Feature	Intel i486	Motorola MC68040	NS 32532
Instruction set Size & Word length	157 instructions, 32 bits	113 instructions, 32 bits	63 instructions, 32 bits.
Addressing Modes	12	18	9
Integer Unit & GPR's	32-bit ALU with 8 registers	32-bit ALU with 16 registers	32-bit ALU with 8 registers
On-Chip Caches & MMUs	8-KB unified cache for both code & data	4-KB Codes cache 4-KB data cache with separate MMU's	512-B Code Cache 1-KB data Cache.

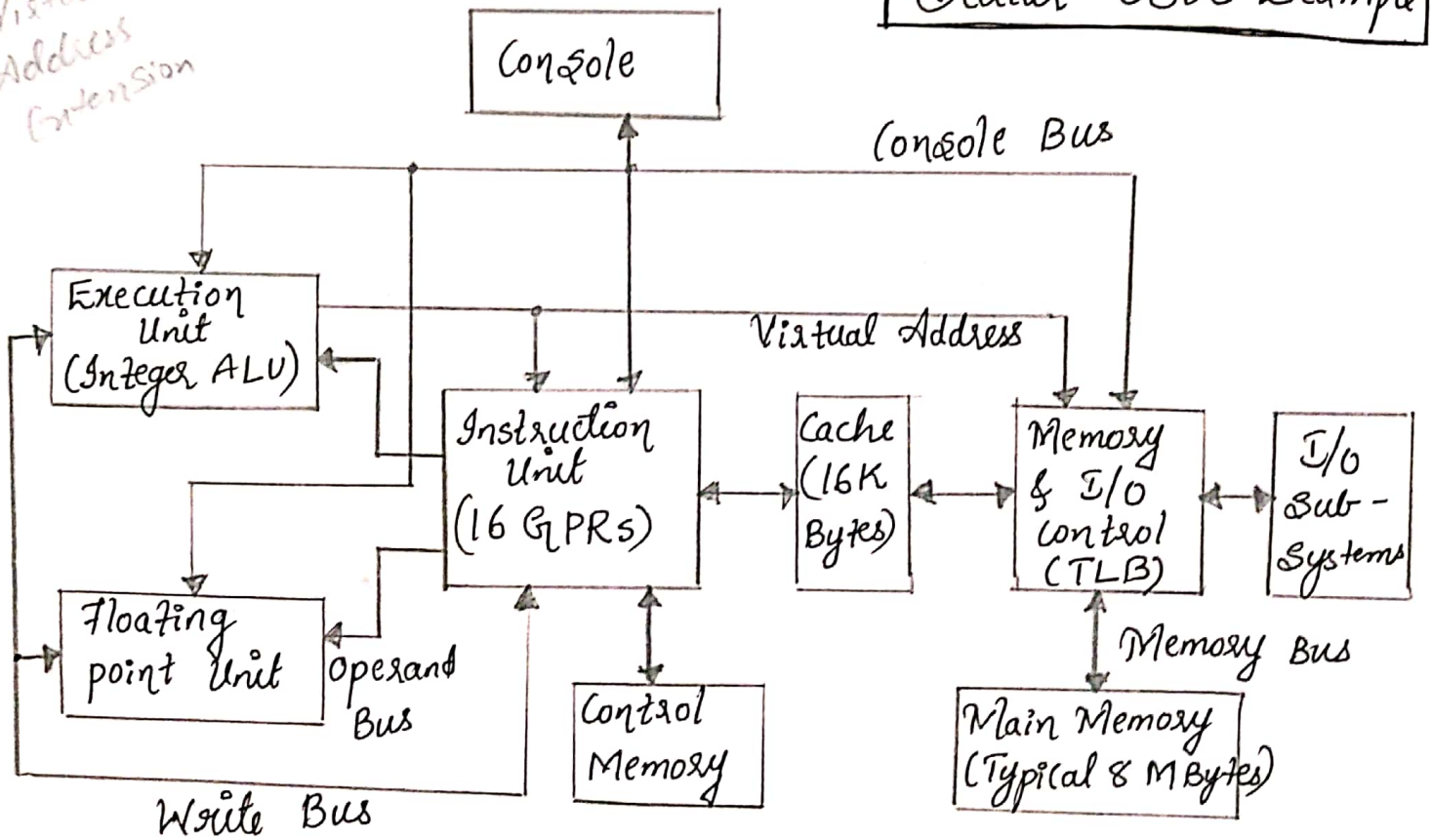
Feature	Intel i486	Motorola MC68040	NS 32532
Floating-point unit, registers, & function units	On-chip with 8 FP registers adder, multiplier, Shifter	On-chip with 3 pipeline stages, & 80-bit FP registers	Off-chip FPU NS 32381, or WTL 3164
Pipeline Stages	5	6	4
Protection level	4	2	2
Memory Organization & TLB/ATC entries	Segmented paging with 4KB/page & 32 entries in TLB	Paging with 4 or 8 KB/page, 64 entries in each ATC	Paging with 4 KB/page, 64 entries
Technology clock rate, packaging, & year introduced	CHMOS IV, 25MHz, 33MHz, 1.2M transistors, 168 pins, 1989.	0.8-μm HCMOS, 1.2M transistors 20MHz, 40MHz, 179 pins, 1990	1.25μm CMOS 370K transistors, 30MHz, 175 pins, 1987
Claimed performance	24 MIPS at 25MHz	20 MIPS at 25MHz, 30 MIPS at 60MHz	15 MIPS at 30MHz
Successors to watch	i586, i686	MC68050, MC68066	Unknown

- 3 Representative CISC Scalar processors are listed above.
- The VAX8600 processor is built on a PC board.
- The i486 & M68040 are single-chip microprocessors.
- In any processor design, the designer attempts to achieve higher throughput in the processor pipelines.

- Due to the complexity involved in a CISC scalar processor, the most difficult task for a designer is to shorten the clock cycle to match the simple operation latency.
- This problem is easier to overcome with a RISC architecture.

EXAMPLE 1 OF DIGITAL EQUIPMENT VAX8600 PROCESSOR ARCHITECTURE

Scalar CISC Example



CPU - Central Processor Unit
TLB - Translation Lookaside Buffer
GPR - General Purpose Register

- This machine implements a typical CISC architecture with microprogrammed control.
- The instruction set contains about 300 instructions with 20 different addressing modes.

- VAX 8600 as shown in fig executes the same instruction set, runs the same VMS operating system, & interfaces with the same I/O buses as VAX 11/780.
- The CPU in the VAX 8600 consists of 2 functional units for concurrent execution of integer & floating-point instructions.
- The unified cache is used for holding both instructions & data.
- There are 16 GPR's in the instruction unit.
- Instruction pipeling has been built with six stages in VAX 8600.
- The instruction unit prefetches & decodes instructions, handles branching operations & supplies operands to the 2 functional units in a pipelined fashion.
- A translation lookaside buffer (TLB) is used in the memory control unit for fast generation of a physical address from a virtual address.
- Both integer & floating-point units are pipelined.
- The performance of the processor pipelines relies heavily on the cache hit ratio & on minimal branching damage to the pipeline flow.
- The CPI of a VAX 8600 instruction varies within a wide range from 2 cycles to as high as 20 cycles.

Example: Both multiply & divide may tie up the execution unit for a large number of cycles.

- This is caused by the use of long sequences of microinstructions to control H/w operations.
- The general philosophy of designing a CISC processor is to implement useful instructions in H/w/firmware which may result in a shorter program length with a lower S/w overhead.
- This advantage has been obtained at the expense of a lower clock rate & a higher CPI which may not pay off at all.
- The VAX8600 was improved from the earlier VAX/11 Series.
- The System was later further upgraded to the VAX 9000 Series offering both vector H/w & multiprocessor options.
- All the VAX Series have used a paging technique to allocate the physical memory to user programs.

CISC Microprocessor Families

- The Intel 4004 appeared as 1st microprocessor based on a 4-bit ALU, then 8 bit 8008, 8080, & 8085, 16 bit 8086, 8088, 80186 & 80286, 32-bit 80386, 80486 & 80586 are the latest 32-bit micros in Intel 80x86 family.
- Motorola produced its first 8-bit micro, MC6800, then 16-bit 68000, 32-bit 68020, MC68030 & MC68040 are the latest Motorola MC680x0 family.

Example 2 : The Motorola MC68040 microprocessor Architecture

Scalar CISC Family

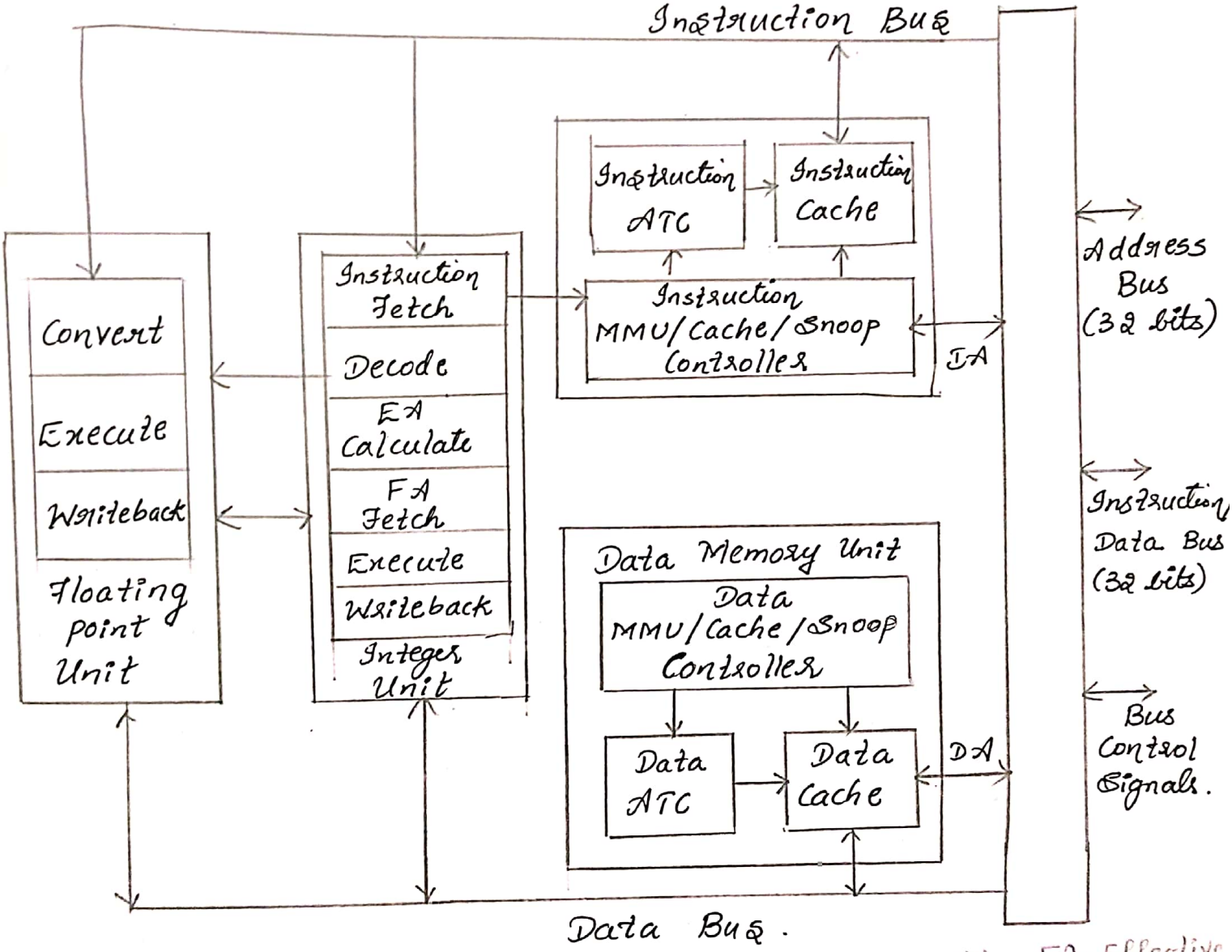


Fig: Architecture of MC68040 processor

EA - Effective Address
FA - Final Address

(High-density CMOS)

- The MC68040 is a 0.8μm HCMOS microprocessor containing more than 1.2 million transistors, comparable to the i80486.
- Fig shows the MC68040 architecture.
- The processor implements over 100 instructions using 16 GPR, a 4-K byte data cache, & a 4-K byte instruction cache, with separate memory management units (MMUs) supported by an address translation cache (ATC)

which is equivalent to the TLB used in other systems.

- The data formats range from 8 to 80 bits, based on IEEE floating-point standard.
- 18 Addressing modes are supported, including register direct & indirect, indexing, memory indirect, program counter indirect, absolute & immediate modes.
- The instruction set includes data movement, integer, BCD, & floating point arithmetic, logical, shifting, bit-field manipulation, cache maintenance, & multiprocessor communications, in addition to program & system control & memory management instructions.
- The integer unit is organized in a Six-Stage instruction pipeline.
- The floating point unit consists of 3 pipeline stages.
- All instructions are decoded by the integer unit.
- Floating point instructions are forwarded to the floating-point unit for execution.
- Separate instruction & data buses are used to & from the instruction & data memory units, respectively.
- Dual MMU's allow interleaved fetch of instructions & data from the main memory.
- Both the address bus & the data bus are 32 bits wide.
- 3 Simultaneous memory requests can be generated by the dual MMU's including data operand read & write & instruction pipeline refill.

- Snooping logic is built into the memory unit for monitoring bus events for cache invalidation.
- The complete memory management is provided with a virtual demand paged OS.
- Each of the 2 ATC's has 64 entries providing fast translation from virtual address to physical address.
- With the CISC complexity involved, the M68040 does not provide delayed branch H/w support, which is often found in RISC processors like Motorola's M88100 microprocessor.

II RISC Scalar Processors

- Generic RISC processors are called Scalar RISC because they are designed to issue one instruction per cycle, similar to the base scalar processor shown in ideal case of pipelinge.
- In theory both RISC & CISC Scalar processors should perform about the same if they run with the same clock rate & with equal program length.
- However these 2 assumptions are not always valid as the architecture affects the quality & density of code generated by compilers.
- The RISC design gains its power by pushing some of the less frequently used operations into S/w.
- The reliance on a good compiler is much more demanding in a RISC processor than in a CISC processor.

- Instruction-level parallelism is exploited by pipelining in both processor architecture.
- Without a high clock rate, a low CPI, and good compilation support, neither CISC nor RISC can perform well as designed.
- The simplicity introduced with a RISC processor may lead to the ideal performance of the base scalar machine modeled in ideal pipelining case.

Representative RISC Scalar Processors

Feature	Sun SPARC CY7601	Intel i860	Motorola M 88100	AMD 29000
Instruction set, formats, addressing modes	69 Inst., 32-bit format, 7 data types, 4-stage instr, pipeline	82 inst, 32-bit format, 4 addressing modes	51 insts. 7 data types, 3 instr. formats, 4 addressing modes	112 insts, 32 bit format, all registers indirect addressing
Integer unit, GPRs	32-bit RISC, 36 registers divided into 8 windows	32-bit RISC Core, 32 registers.	32-bit IU with 32 GPRs & Scoreboarding	32-bit IU with 192 registers without windows.
Cache(s), MMU & memory organization	Off-chip cache/ MMU on CY7C604 with 64-entry TLB.	4-KB code, 8 KB data, On-chip MMU, paging with 4 KB/page	Off-chip M88200 Caches/MMUs, Segmented paging, 16-KB cache.	On-chip MMU with 32-entry TLB with 4-word prefetch buff & 512-B branch targ Cache.
FPU registers & fundions	Off-chip FPU on CY7C602, 32 registers, 64-bit pipeline	On-chip 64-bit FP multiplier & FP Adder with 32 FP, 3-D Graphics unit	On-chip FPU Adder	Off-chip FPU on AMD 2902

Features	Sun SPARC CY7C601	Intel i860	Motorola M 88100	AMD 29000
Operation modes	Concurrent IU & FPU operations	Allow dual insts & dual FP operations	Concurrent IU, FPU & memory access with delayed branch	4-Stage pipeline processed
Technology, clock rate, packaging, & year	0.8 µm CMOS IV, 33 MHz, 207 pins, 1989	1-µm. CHMOS IV over 1M transistors, 40MHz, 168 pins 1989	1-µm HCMOS, 1.2M transistors, 20 MHz, 180 pins, 1988	1.2 µm CMOS, 30 MHz, 40MHz, 169 pins 1988
Claimed performance & Successor to watch	24 MIPS for 33 MHz version, 50 MIPS for 80 MHz ECL version, Up to 32 register window can be built	40 MIPS & 60 M Flops for 40 MHz, i860/XP announced in 1992 with 2.5M transistor	17 MIPS & 6 M Flops at 20 MHz up to 7 special function units can be configured.	27 MIPS at 40 MHz, new version AMD 29050 at 55 MHz in 1990

Representative RISC Processors

- 4 RISC-based processors, the Sun SPARC, Intel i860, Motorola M88100, & AMD 29000, are summarized in the table above.
- All of these processors use 32-bit instr's.
- The instruction sets consist of 51 to 124 basic inst's.
- On chip floating-point units are built into the i860 & M88100, while SPARC & AMD use off-chip floating point units

- We consider these 4 processors generic scalar RISC issuing essentially only one instruction per pipeline cycle.
- We examine Sun SPARC & i860 architectures.
- SPARC stands for scalable processor architecture.
- The scalability of the SPARC architecture refers to the use of a different number of register windows in different SPARC implementations.
- This is different from the M88100, where scalability refers to the number of special function units (SFUs) implementable on different versions of M88000 processor.
- The Sun SPARC is derived from original Berkeley RISC design.

Example 1: The Sun Microsystems SPARC architecture

- The SPARC has been implemented by a number of licensed manufacturers as given in table below.
- Different technologies & window numbers are used by different SPARC manufacturers.

SPARC Chip	Technology	Clock Rate (MHz)	Claimed VAX MIPS
Cypress CY7C601 IU	0.8 μm CMOS IV, 207 pins	33	24

SPARC Chip	Technology	Clock Rate (MHz)	Claimed VAX MIPS
Fujitsu MB 86901 IU	1.2 μm CMOS, 179 pins	25	15
LSI Logic L64811	1.0 μm HCMOS, 179 pins	33	20
TI 8846	0.8 μm CMOS	33	24
BEIU B-3100	ECL Family	80	50

- All of these manufacturers implement the floating-point unit (FPU) on a separate co-processor chip.
- The SPARC processor architecture contains essentially a RISC integer unit (IU) implemented with 2 to 32 register window.
- SPARC family chips produced by Cypress Semiconductor Shown in fig below is the architecture of the Cypress CY7C601 SPARC processor & of CY7C602 FPU.
- The Sun SPARC instruction set contains 69 basic instructions, a significant increase from 39 instructions in the original Berkeley RISC II instruction set.

Fig: The SPARC architecture with processor & FPU on 2 Separate chips

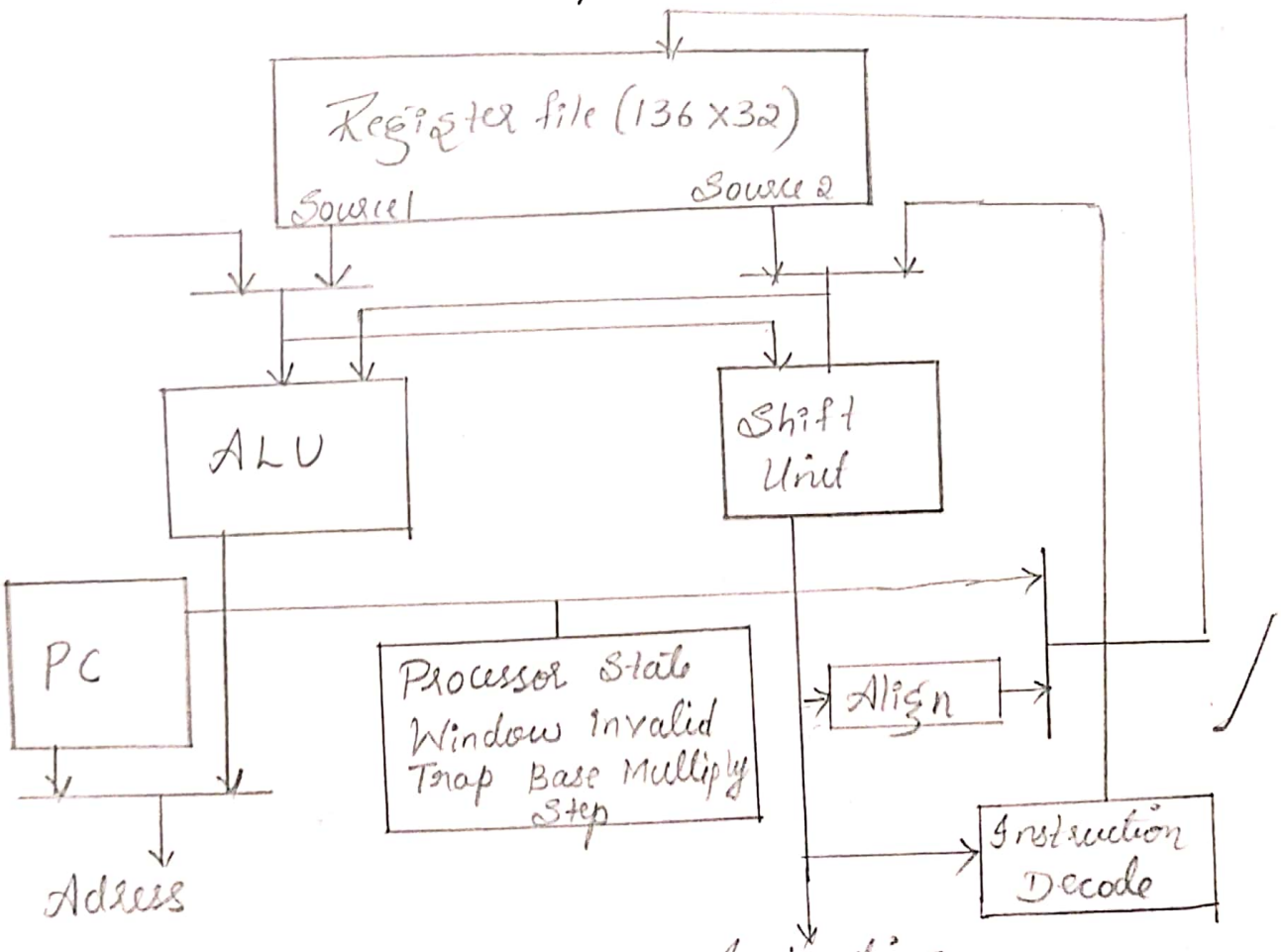


Fig (a) The Cypress CY7C601 SPARC processor

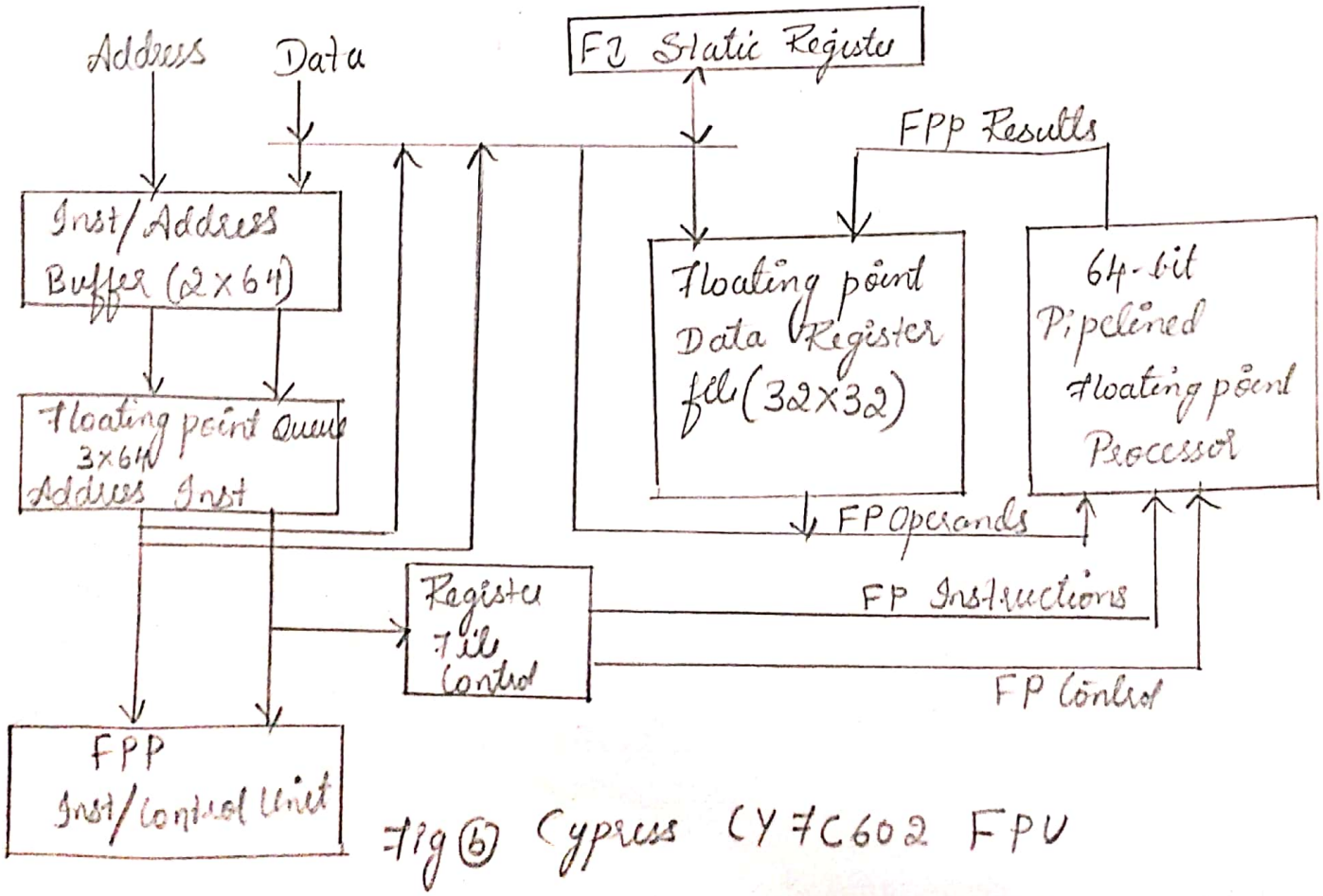


Fig (b) Cypress CY7C602 FPU

- The SPARC runs each procedure with a set of thirty two 32-bit IU registers.
- Eight of these registers are global registers shared by all procedures, & the remaining 24 are window registers associated with only each procedure.
- The concept of using overlapped register windows is the most important feature introduced by Berkeley RISC architecture.

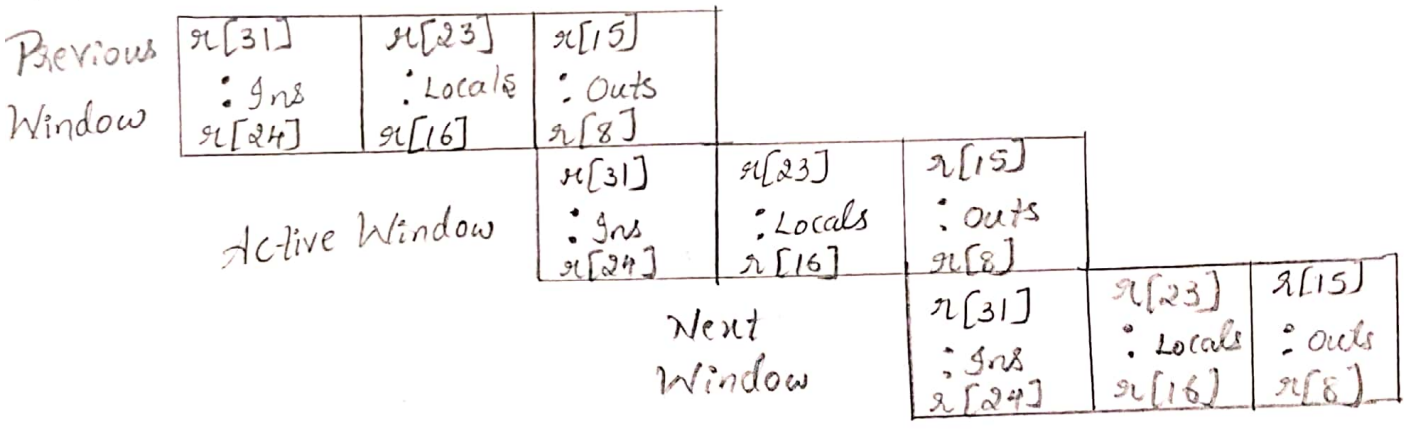


Fig: 3 Overlapping register windows & the global registers

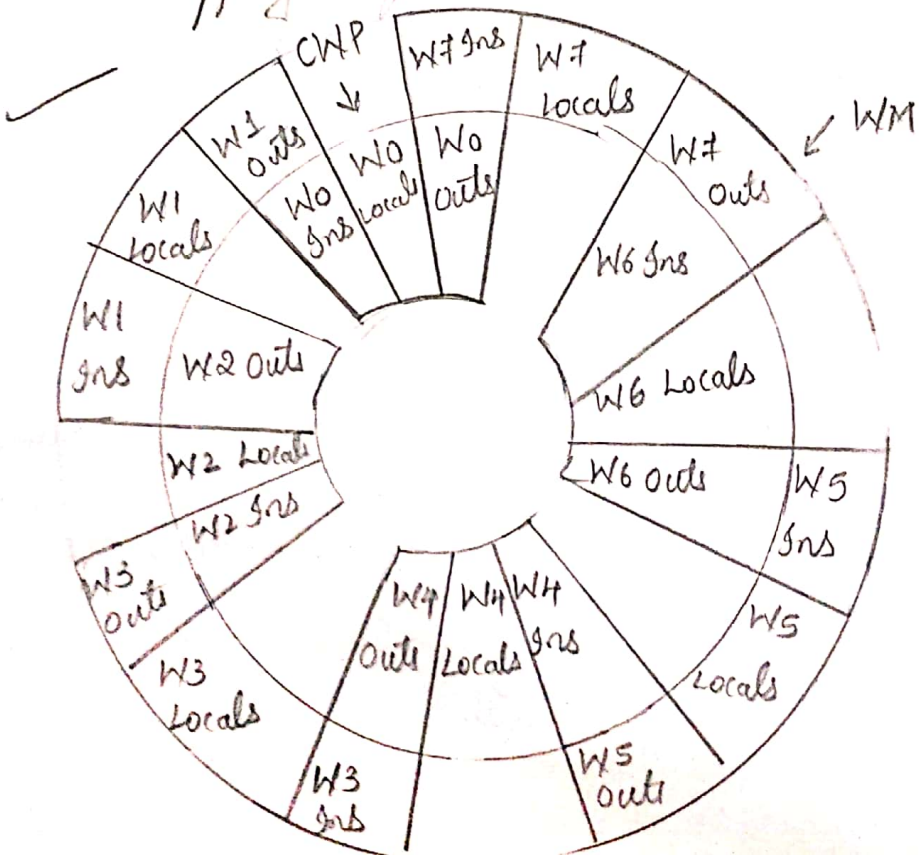


Fig: 8 register windows forming a circular clock.

- The concept is illustrated in fig for 8 overlapping Windows (formed with 64 local registers & Overlapped registers) & eight globals with a total of 136 registers as implemented in the Cypress 601.
- Each register Window is divided into three 8-register Sections, labeled Ins, Locals, & Outs.
- The Local registers are only locally addressable by each procedure.
- The Ins & Outs are Shared among procedures.
- The calling procedure passes parameters to the called procedure via its Outs (r8 to r15) registers, which are the Ins registers of the called procedure.
- The window of the currently window pointer.
- A window invalid mask is used to indicate which window is invalid.
- The trap base-register serves as a pointer to a trap handler.
- A Special register is used to create a 64-bit product in multiple step instructions.
- Procedures can also be called without changing the window.
- The overlapping windows can Significantly Save the time required for interprocedure communications, resulting in much faster context switching among co-operative procedures.
- The FPU features 32 single-precision (32-bit) or 16 double-precision (64-bit) floating point operations.

from fig (b).

- Fourteen of the 69 SPARC instructions are for floating-point operations.
- The SPARC architecture implements 3 basic instruction formats all using a single word length of 32 bits.

Example 2: Intel i860 processor architecture

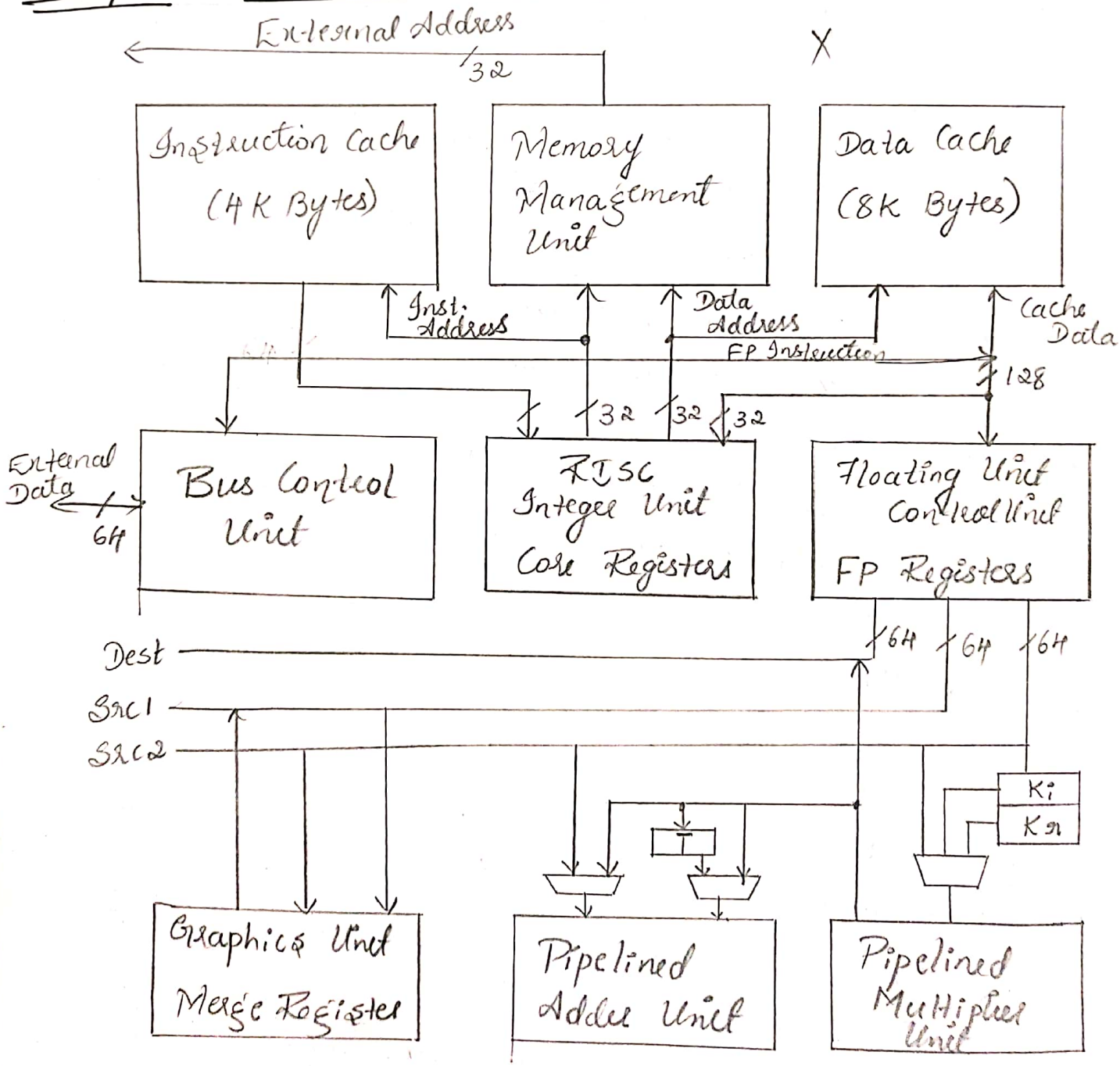
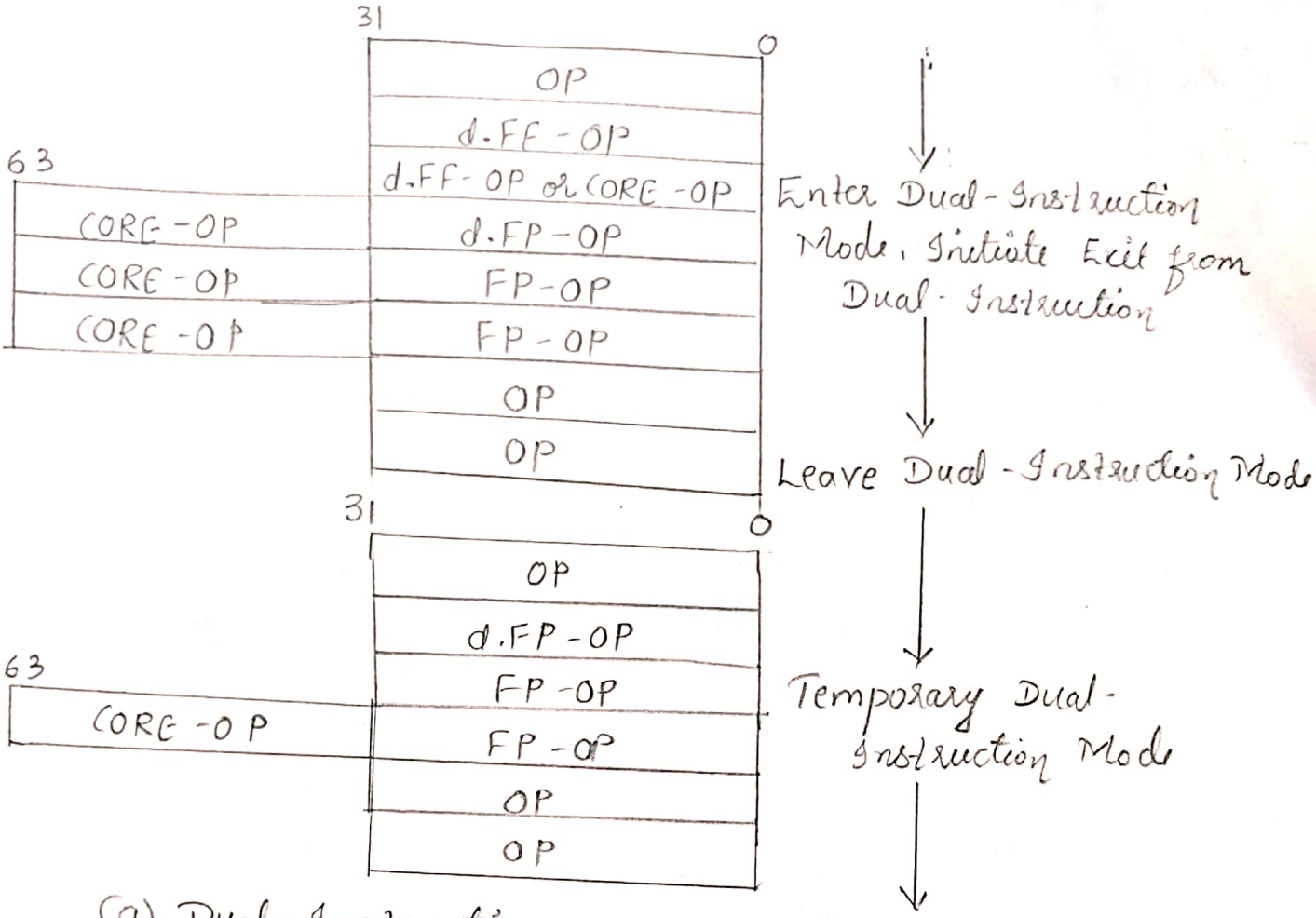


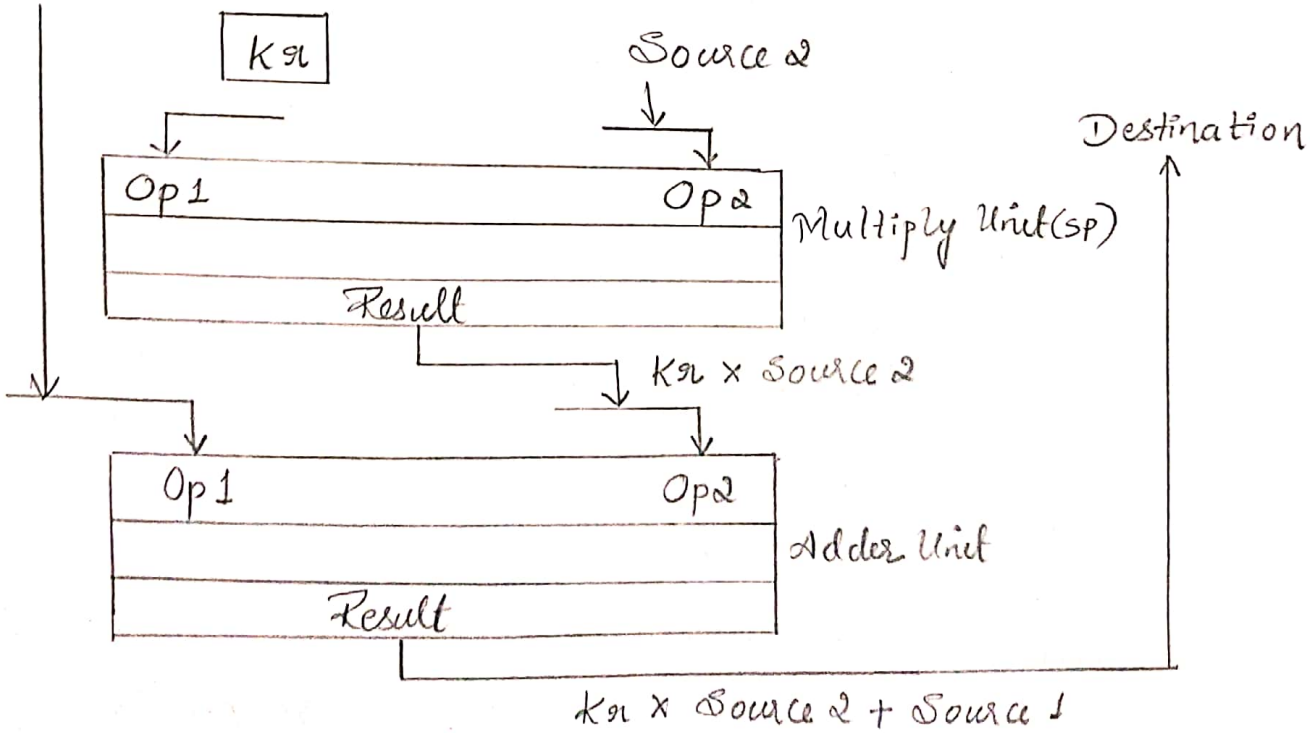
Fig: Functional Units & data paths of the Intel i860 RISC microprocessor.

- In 1987, Intel Corporation introduced the i860 microprocessor.
- It is a 64-bit RISC processor fabricated on a single chip containing more than 1 million transistors.
- The peak performance of the i860 was designed to achieve 80 Mflops single precision or 60 Mflops double precision, or 40 MIPS in 32-bit integer operations at a 40-MHz clock rate.
- A schematic block diagram of major components in the i860 is shown in fig above.
- There are 9 functional units (shown in 9 boxes) interconnected by multiple data paths with widths ranging from 32 to 128 bits.
- All external or internal address buses are 32-bit wide, & the external data path or internal data bus is 64 bits wide.
- However the internal RISC integer ALU is only 32 bits wide.
- The instruction cache has 4 Kbytes organized as a 2 way set-associative memory with 32 bytes per cache block.
- It transfers 64 bits per clock cycle, equivalent to 320 Mbytes/s at 40 MHz.
- The data cache is a 2 way set-associative memory of 8 Kbytes.
- It transfers 128 bits per clock cycle (640 Mbytes/s) at 40 MHz.

- A write-back policy is used Cacheing can be inhibited by s/w, if needed.
- The bus control unit coordinates the 64-bit data transfer bet" the chip & the outside world.
- The MMU implements protected 4K byte paged virtual memory of 2^{32} bytes via of TLB.
- The paging & MMU structure of the i860 is identical to that implemented in the i486.
- An i860 & an i486 can be used jointly in a heterogeneous multiprocessor system, permitting the development of compatible OS kernels.
- The RISC integer unit (IU) execute load, store, integer, bit, & control instructions & fetches instructions for the floating-point control unit as well.
- There are 2 floating-point units, namely, the multiplier unit & the adder unit which can be used separately or simultaneously under the co-ordination of the floating point control unit.
- Special dual-operation floating-point instructions such as add-and-multiply & subtract-and-multiply use both the multiplier & adder units in 11e1 (fig below)
- Both the integer unit & floating-point control unit can execute concurrently.
- The i860 is also a superscalar RISC processor capable of executing 2 instructions, one integer & one floating point at the same time.



(a) Dual-Instruction mode transitions.



(b) Dual operations in floating point units.

- The floating point unit conforms to the IEEE 754 floating-point standard, operating with single-precision (32-bit) & double-precision (64-bit) operands.
- The graphics unit executes integer operations corresponding to 8, 16, 32 bit pixel data types.
- This unit supports 3 dimensional drawing in a graphics frame buffer with color intensity, shading & hidden surface elimination.
- The merge register is used only by vector integer instructions.
- This register accumulates the results of multiple addition operations.
- The i860 executes 82 instructions including 42 RISC integer, 24 floating point 10 graphics & 6 assembler pseudo operations.
- All the instructions executes in one cycle, which equals 25ns for a 40-MHz clock rate.
- The 1860 & its successor, the i860XP are used in floating point accelerators, graphics subsystems, workstations, multiprocessors & multicomputers.

RISC IMPACTS

- RISC will outperform CISC if the program length does not increase dramatically.
- Based on one reported experiment, converting from a CISC program to an equivalent RISC program increases the code length by only 40%.

- The increase depends on program behavior, & 40% increase may not be typical of all programs.
- Nevertheless, the increase in code length is much smaller than the increase in clock rate & reduction in CPI.
- Thus the intuitive reasoning should previal in both cases.
- A RISC processor lacks some sophisticated instructions found in CISC processors.
- The increase in RISC program length implies more instruction traffic & greater memory demand.
- Another RISC problem is caused by the use of a large register file.
- Although a larger register set can hold more intermediate results & reduce the data traffic betn the CPU & memory, the register decoding system will be more complicated.
- Longer register access time places a greater demand on the compiler to manage the register window functions.
- Another shortcoming of RISC lies in its hardwired control, which is less flexible & more error-prone.
- RISC shortcomings are directly related to some of its claimed advantages.
- More benchmark & application experiments are needed to determine the optimal sizes of the register set, I-cache, & D-cache.

- Further processor improvements may include a 64-bit integer ALU, multiprocessor support such as Snoopy logic for cache coherence control, faster interprocessor synchronization or H/w support for message passing, & special-function units for I/O interface & graphics support.
- The boundary bet^n RISC & CISC architectures has become blurred because both are now implemented with the same H/w technology.

Example: The VAX 9000, Motorola 88100, & i586 are built with mixed features taken from both the RISC & CISC camps.

- It is the applications that eventually determine the best choice of a processor architecture.

Superscalar & Vector Processors

- A CISC or a RIS scalar processor can be improved with a superscalar or Vector architecture.
- Scalar processors are those executing one instruction per cycle.
- Only one instruction is issued per cycle, & only one completion of instruction is expected from the pipeline per cycle.
- In a superscalar processor, multiple instruction pipelines are used.
- This implies that multiple instructions are issued per

- The floating point unit conforms to the IEEE 754 floating-point standard, operating with single-precision (32-bit) & double-precision (64-bit) operands.
- The graphics unit executes integer operations corresponding to 8, 16, 32 bit pixel data types.
- This unit supports 3 dimensional drawing in a graphics frame buffer with color intensity, shading & hidden surface elimination.
- The merge register is used only by vector integer instructions.
- This register accumulates the results of multiple addition operations.
- The i860 executes 82 instructions including 42 RISC integer, 24 floating point 10 graphics & 6 assembler pseudo operations.
- All the instructions executes in one cycle, which equals 25ns for a 40-MHz clock rate.
- The 1860 & its successor, the i860XP are used in floating point accelerators, graphics subsystems, workstations, multiprocessors & multicomputers.

RISC IMPACTS

- RISC will outperform CISC if the program length does not increase dramatically.
- Based on one reported experiment, converting from a CISC program to an equivalent RISC program increases the code length by only 40%.

- The increase depends on program behavior, & 40% increase may not be typical of all programs.
- Nevertheless, the increase in code length is much smaller than the increase in clock rate & reduction in CPI.
- Thus the intuitive reasoning should previal in both cases.
- A RISC processor lacks some sophisticated instructions found in CISC processors.
- The increase in RISC program length implies more instruction traffic & greater memory demand.
- Another RISC problem is caused by the use of a large register file.
- Although a larger register set can hold more intermediate results & reduce the data traffic betn the CPU & memory, the register decoding system will be more complicated.
- Longer register access time places a greater demand on the compiler to manage the register window functions.
- Another shortcoming of RISC lies in its hardwired control, which is less flexible & more error-prone.
- RISC shortcomings are directly related to some of its claimed advantages.
- More benchmark & application experiments are needed to determine the optimal sizes of the register set, I-cache, & D-cache.

- Further processor improvements may include a 64-bit integer ALU, multiprocessor support such as Snoopy logic for cache coherence control, faster interprocessor synchronization or H/w support for message passing, & special-function units for I/O interface & graphics support.
- The boundary bet^n RISC & CISC architectures has become blurred because both are now implemented with the same H/w technology.

Example: The VAX 9000, Motorola 88100, & i586 are built with mixed features taken from both the RISC & CISC camps.

- It is the applications that eventually determine the best choice of a processor architecture.

Superscalar & Vector Processors

- A CISC or a RIS scalar processor can be improved with a superscalar or Vector architecture.
- Scalar processors are those executing one instruction per cycle.
- Only one instruction is issued per cycle, & only one completion of instruction is expected from the pipeline per cycle.
- In a superscalar processor, multiple instruction pipelines are used.
- This implies that multiple instructions are issued per

cycle. & multiple results are generated per cycle.
- A vector processor executes vector instructions on array of data.
- Thus each instruction involves a string of repeated operations which are ideal for pipelining with one result per cycle.

III

Superscalar Processors

- Superscalar processors are designed to exploit more instruction-level parallelism in user programs.
- Only independent instructions can be executed in parallel without causing a wait state.
- The amount of instruction-level parallelism varies widely depending on the type of code being executed.
- It has been observed that the average value is around 2 for code without loop unrolling.
- ∴ for these codes there is not much benefit gained from building a machine that can issue more than 3 instructions per cycle.
- The instruction-issue degree in a superscalar processor has thus been limited to 2 to 5 in practice.

Pipelining in Superscalar Processors

- The fundamental structure of a superscalar pipeline is illustrated in fig below.
- The diagram shows the use of 3 instruction pipeline in IIC for a triple issue processor.

- Superscalar processors were originally developed as an alternative to vector processors.

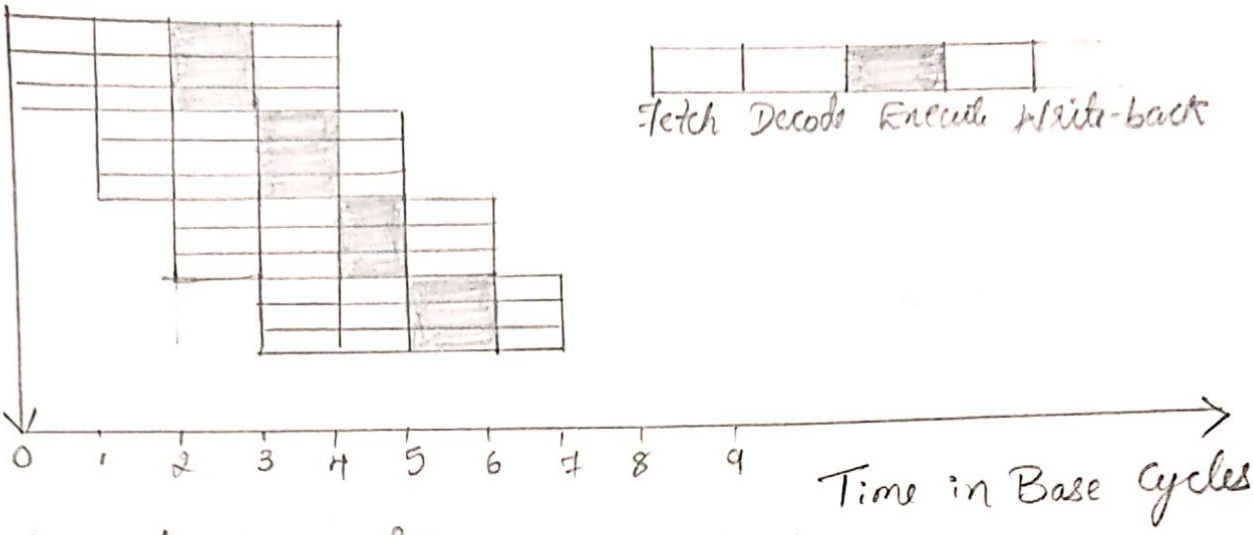
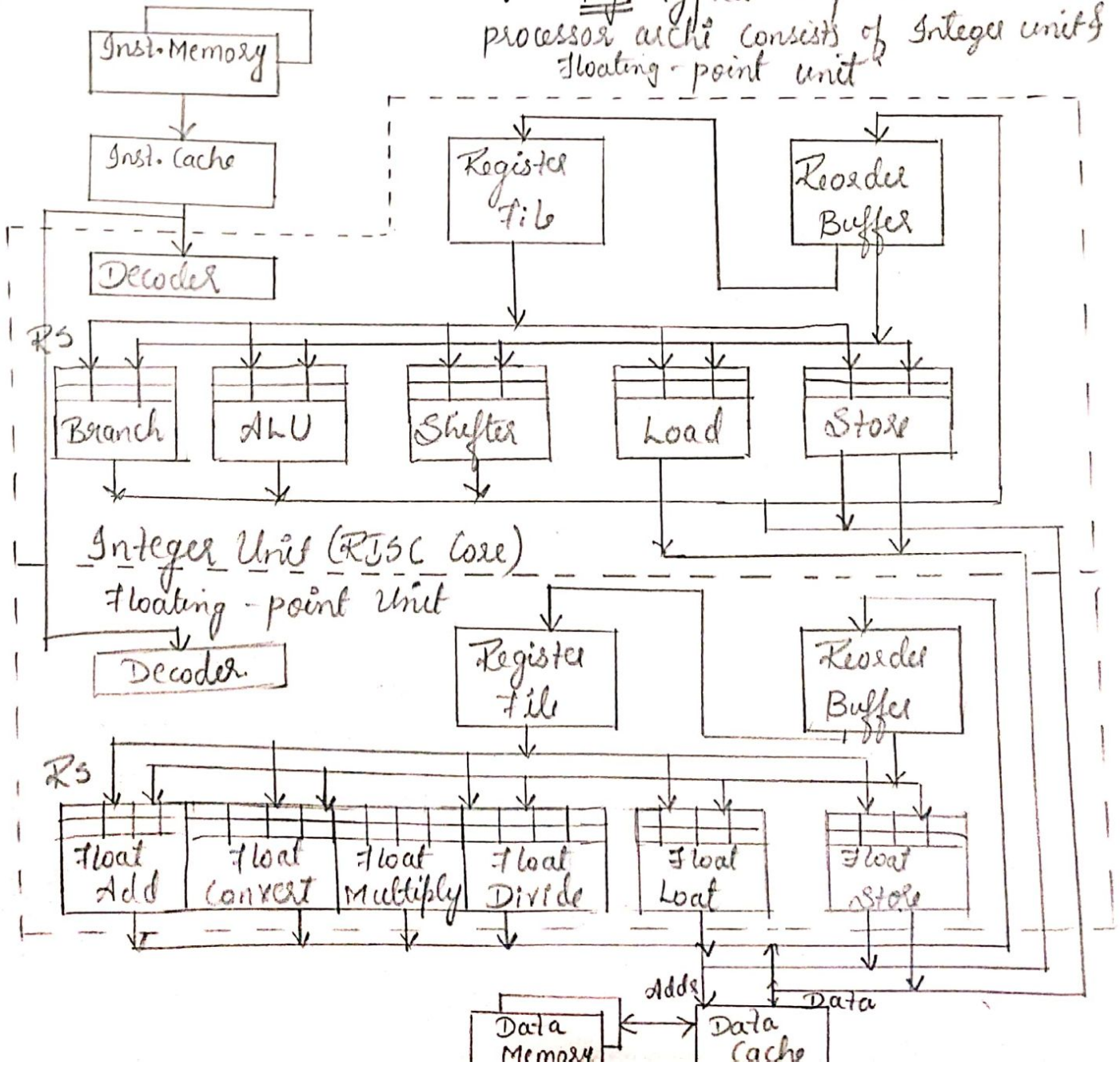


Fig: A Superscalar processor of degree $m = 3$

- A Superscalar processor of degree m can issue m instructions per cycle.
- In this sense, the base scalar processor, implemented either in RISC or CISC has $m = 1$.
- In order to fully utilize a superscalar processor of degree m, m instructions must be executable in parallel. This situation may not be true in all clock cycles.
- Some of the pipelines may be stalling in a wait state.
- In a Superscalar processor, the simple operation latency should require only one cycle, as in base scalar processor.
- Due to the desire for a higher degree of Instruction level parallelism in programs, the superscalar processor depends more on an optimizing compiler to exploit parallelism.

- In theory, a superscalar processor can attain the same performance as a machine with vector H/w.
- A superscalar machine that can issue a fixed-point, floating-point, load, & branch all in one cycle achieves the same effective parallelism as a vector machine which executes a vector load, chained into a vector add, with one element loaded & added per cycle.
- ~~This will become more evident~~.
- A typical superscalar architecture for a RISC processor is shown in fig below.

Fig: Typical Superscalar RISC processor archi consists of Integer unit & Floating-point unit



- Multiple instruction pipelines are used.
- The instruction cache supplies multiple instructions per fetch.
- However, the actual number of instructions issued to various functional units may vary in each cycle.
- The number is constrained by data dependences & resource conflicts among instructions that are simultaneously decoded.
- Multiple functional units are built into the integer unit & into the floating-point unit.
- Multiple data buses exist among the functional units.

Representative Superscalar Processors

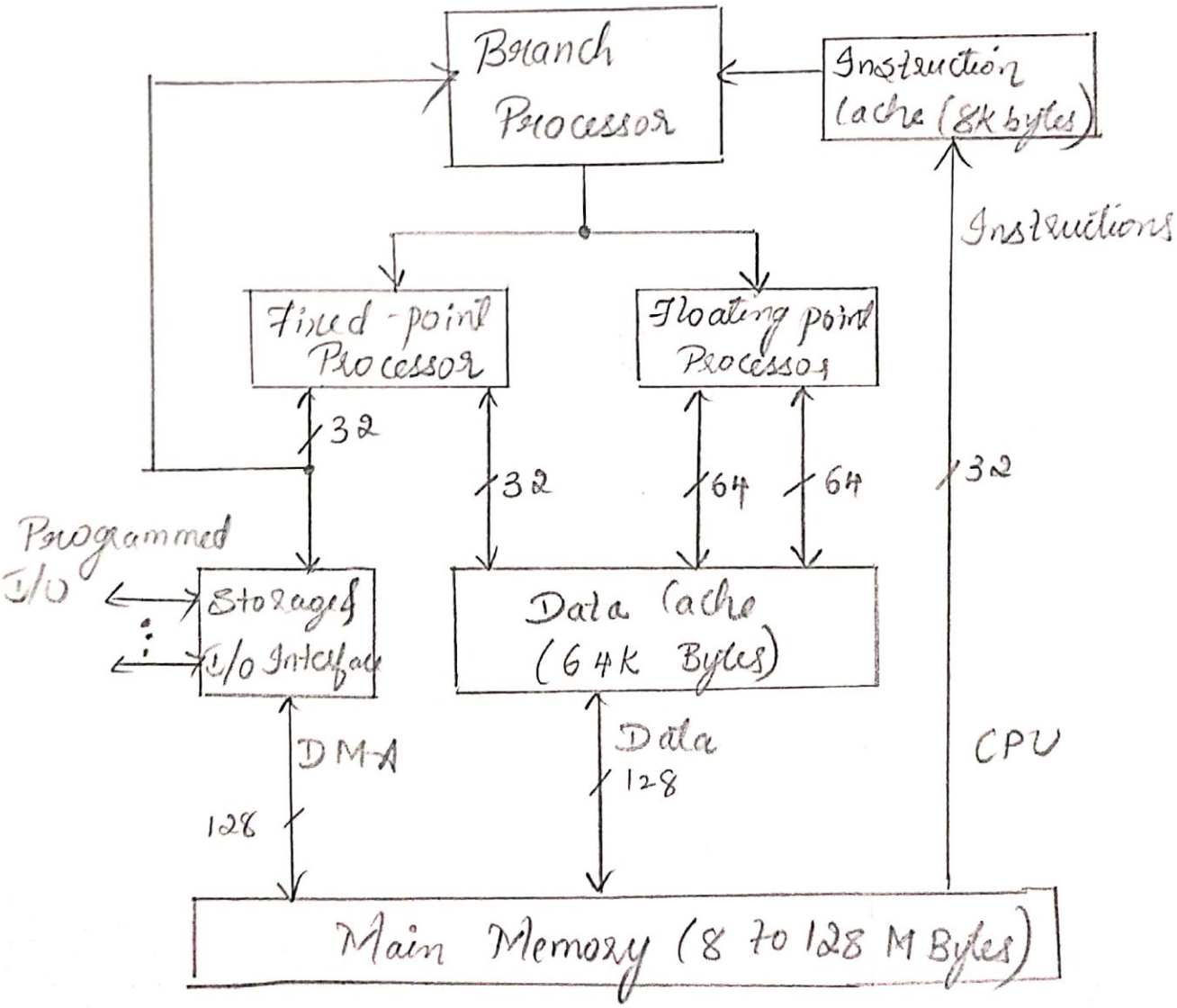
- A number of commercially available processors have been implemented with the Superscalar architecture.
- Notable ones include the IBM RS/6000, DEC 21064, & intel i960CA processors as summarized in table
- Due to reduced CPI & higher clock rates used, most superscalar processors out-perform Scalar processors.
- The maximum number of instructions issued per cycle ranges from 2 to 5 in these 4 Superscalar processors.
- Typically the register files in IU & FPU each have 32 registers.
- Most Superscalar degree is low due to limited instruction parallelism that can be exploited in ordinary programs.

Representative Superscalar Processors

Feature	Intel i960CA	IBM RS/6000	DEC 21064
Technology, Clock rate, year	20MHz, 1986	1-μm CMOS, 30 MHz, 1990	0.75μm CMOS, 150 MHz, 431 pins, 1992.
Functional Units & multiple Instruction issue	Issue up to 3 instructions (register, memory & control) per cycle, Seven functional units available for concurrent use	POWER archi, issue 4 inst's (1FXU, 1FPU, & 2 ICU operations) per cycle	Alpha archi, issue 2 inst's per cycle, 64-bit IU & FPU, 128-bit data bus & 34-bit address bus implemented in initial version
Registers, Cache, MMU, address Space.	1 KB, I-Cache, 1.5-KB RAM, 4-channel I/O with DMA, 11th decode, multiported registers	32 32-bit GPRs, 8-KB I-Cache, 64 KB D-Cache with separate TLBs.	32 64 bit GPRs, 8-KB I Cache, 8 KB D Cache, 64-bit virtual Space designed, 43 bit address Space implemented in initial version
Floating point Unit & functions	On-chip FPU, fast multimode interrupt, multitask	On-chip FPU 64-bit multiply, add, divide, Subtract, IEEE 754 Standard.	On-chip FPU, 32 64-bit FP registers, 10-Stage pipeline, IEEE & VAX FP Standards
Claimed performance & remarks.	30 VAX/MIPS peak at 25MHz, real-time embedded system Control & multiprocessor applications.	34 MIPS & 11 M flops at 25MHz on POWER Station 530	300 MIPS peak & 150 M flops peak at 150 MHz, multiprocessor & Cache coherence Support

- Besides the register files, reservation stations & reorder buffers can be used to establish instruction windows.
- The purpose is to support instruction lookahead & internal data forwarding, which are needed to schedule multiple instructions through the multiple pipelines simultaneously.

Example 1: IBM RS/6000 architecture.

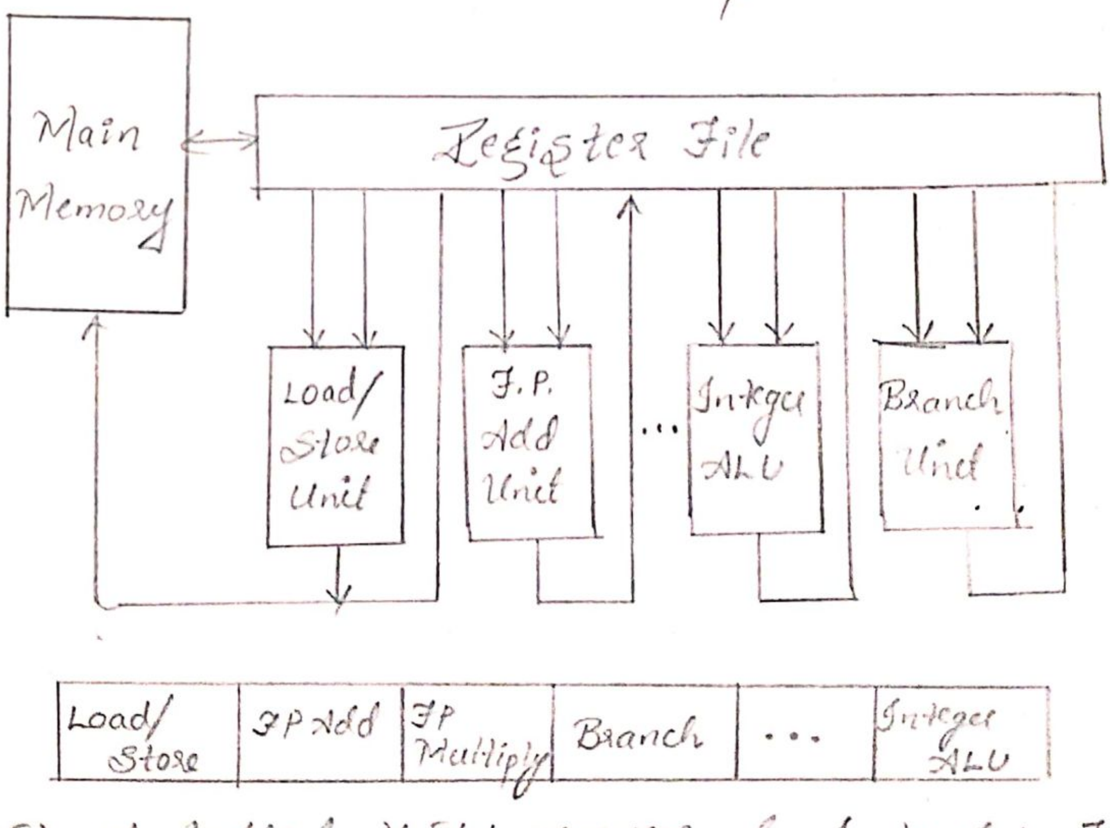


- IBM announced the RISC System 6000.
- It is a Superscalar processor as illustrated in fig above.
- There are 3 functional units called the branch processor, fixed point unit & floating point units, which can operate in ||el.
- The branch processor can arrange the execution of upto 5 instructions per cycle.
- These include one branch instruction in the branch processor, one fixed point instruction in the FXU, one condition register instruction in the branch processor, & one floating point multiply add instruction in the FPU, which can be counted as 2 floating point operations.
- The RS/6000 is hardwired rather than microcoded.
- The System uses a number of wide buses ranging from one word (32 bits) for the FXU to 2 words (64 bits) for the FPU, & 4 words for I-Cache & D-Cache respectively
- These wide buses provide the high instruction & data bandwidths required for Superscalar implementation.
- The RS/6000 design is optimized to perform well in numerically intensive scientific & engineering applications as well as in multiuser commercial environments.
- A number of RS/6000 based workstations & Servers are produced by IBM.

Example: The powerStation 530 has a clock rate of 25 MHz with performances benchmarks reported as 34.5 MIPS & 10.9 M flops.

The VLIW Architecture

- The VLIW architecture is generalized from two well-established concepts
  - Horizontal microcoding → approach for Implementing Processor Control.
  - Superscalar processing. — layer of H/w level of instruc... that implement higher level
- A Typical VLIW (very long Instruction Word) machine has instruction words hundreds of bits in length.
- As illustrated in fig ⓐ below multiple functional units are used concurrently in a VLIW processor.
-



fig(a). A typical VLIW processor & Instruction Format

- All functional units share the use of a common large register file.
- The operations to be simultaneously executed by the functional units are synchronized in a VLIW instruction 256 or 1024 bits per instruction word.

as implemented in the multiflow computer models.

- The VLIW concept is borrowed from horizontal microcoding.
- Different fields of the long instruction word carry the opcodes to be dispatched to different functional units.
- Programs written in conventional short instruction words (say 32 bits) must be compacted together to form the VLIW instructions.
- This code compaction must be done by a compiler which can predict branch outcomes using elaborate heuristics or run-time statistics.

Pipelining in VLIW Processors

-

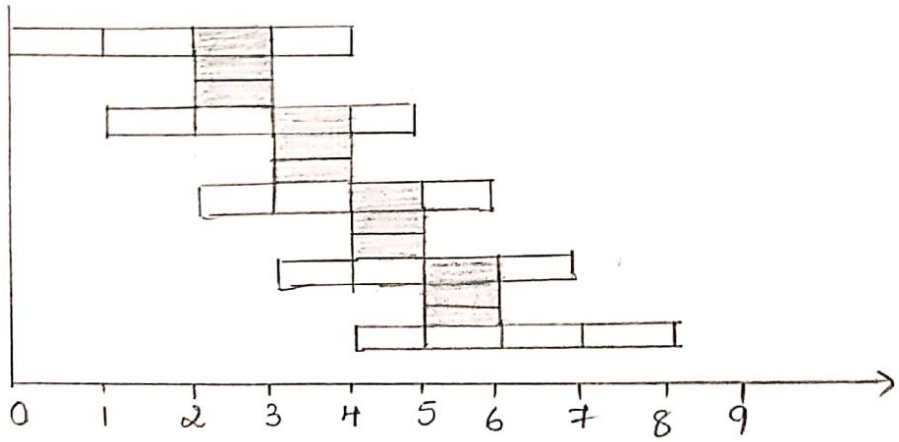


Fig: VLIW execution with degree $n = 3$.

- The execution of instructions by an ideal VLIW processor is shown in above fig.
- Each instruction specifies multiple operations.
- The effective CPI becomes 0.33 in this particular example.

(1/3)

- VLIW machines behave much like Superscalar machines with 3 differences
- First, the decoding of VLIW instructions is easier than that of Superscalar instructions.
- Second, the code density of the Superscalar machine is better when the available instruction-level parallelism is less than that exploitable by the VLIW machine.
- This is because the fixed VLIW format includes bits for nonexecutable operations, while the Superscalar processor issues only executable instructions.
- Third, a Superscalar machine can be object-code-compatible with a large family of nonparallel machines.
- On the contrary, a VLIW machine exploiting different amounts of parallelism would require different instruction sets.
- Instruction parallelism & data movements in a VLIW architecture are completely specified at compile time.
- Run time resource scheduling & synchronization are thus completely eliminated
- One can view a VLIW processor as an extreme of a Superscalar processor in which all independent or unrelated operations are already synchronously compacted together in advance.
- The CPI of a VLIW processor can be even lower than that of a Superscalar processor.

Example: The multiflow Trace computer allows up to Seven operations to be executed concurrently with 256 bits per VLIW instruction.

VLIW Opportunities

- In a VLIW architecture, random parallelism among Scalar operations is exploited instead of regular or Synchronous parallelism as in a vectorized Supercomputer or in an SIMD computer.
- The Success of a VLIW processor depends heavily on the efficiency in code compaction.
- The architecture is totally incompatible with that of any conventional general-purpose processor.
- The instruction parallelism embedded in the compacted code may require a different latency to be executed by different functional units even though the instructions are issued at the Same time.
- ∴ different implementations of the Same VLIW architecture may not be binary-compatible with each other.
- By explicitly encoding parallelism in the long instruction, a VLIW processor can eliminate the H/w or S/w needed to detect parallelism.
- The main advantages of VLIW architecture is its Simplicity in H/w Structure & instruction Set.

- The VLIW processor can potentially perform well in Scientific applications where the program behaviors (branch predictions) in more predictable.
- In general-purpose applications, the architecture may not be able to perform well.
- Due to its lack of compatibility with conventional H/w & S/w, the VLIW architecture has not entered the mainstream of computers.

Vector & Symbolic Processors

- A Vector processor is a co-processor specially designed to perform vector computations.
- A vector Instruction involves a large array of operands.
- In other words, the same operation will be performed over a string of data.
- Vector processors are often used in a multipipelined Supercomputer.
- A vector processor can assume either a register-to-register architecture or memory-to-memory architecture.
- The former uses shorter instructions & vector register files.
- The latter uses memory-based instructions which are longer in length, including memory addresses.

Vector Instructions

- Register based vector instructions appear in most register-to-register vector processors like Cray Supercomputers.
- Denote a vector register of length n as V_i, a Scalar register as S_i, & a memory array of length n as M(1:n).
- Typical register-based vector operations are listed below, where a vector operator is denoted by a small circle "o":

$$V_1 \circ V_2 \longrightarrow V_3 \quad \text{(binary Vector)}$$
$$S_1 \circ V_1 \longrightarrow V_2 \quad \text{(Scaling)}$$
$$V_1 \circ V_2 \longrightarrow S_1 \quad \text{(binary reduction)}$$
$$M(1:n) \longrightarrow V_1 \quad \text{(vector load)}$$
$$V_1 \longrightarrow M(1:n) \quad \text{(vector Store)}$$
$$\circ V_1 \longrightarrow V_2 \quad \text{(unary vector)}$$
$$\circ V_1 \longrightarrow S_1 \quad \text{(unary reduction)}$$

- It should be noted that the vector length should be equal in all operands used in a vector instruction
- The reduction is an operation on one or 2 vector operands, & the result is a Scalar - Such as the dot product betⁿ 2 vectors & the maximum of all components in a vector.

- In all cases, these vector operations are performed by dedicated pipeline units including functional pipelines & memory access pipelines.
- Long vectors exceeding the register length n must be segmented to fit the vector registers n elements at a time.
- Memory based vector operations are found in memory vector processors such as those in Cyber 205.
- Listed below are a few examples

$$M_1(1:n) \circ M_2(1:n) \longrightarrow M(1:n)$$
$$S_1 \circ M_1(1:n) \longrightarrow M_2(1:n)$$
$$\circ M_1(1:n) \longrightarrow M_2(1:n)$$
$$M_1(1:n) \circ M_2(1:n) \longrightarrow M(k)$$

where $M_1(1:n)$ & $M_2(1:n)$ are 2 vectors of length n & $M(k)$ denotes a scalar quantity stored in memory location k.
- Note that the vector length is not restricted by register length.
- Long vectors are handled in a streaming fashion using superwords cascaded from many shorter memory words.

Vector Pipelines

- Vector processors take advantages of unrolled-loop-level parallelism.
- The vector pipelines can be attached to any scalar processor, whether it is superscalar, superpipelined or both.
- Dedicated vector pipelines will eliminate some S/w overhead in looping control, the effectiveness of a vector processor relies on the capability of an optimizing

compiler that vectorizes sequential code for vector pipeline

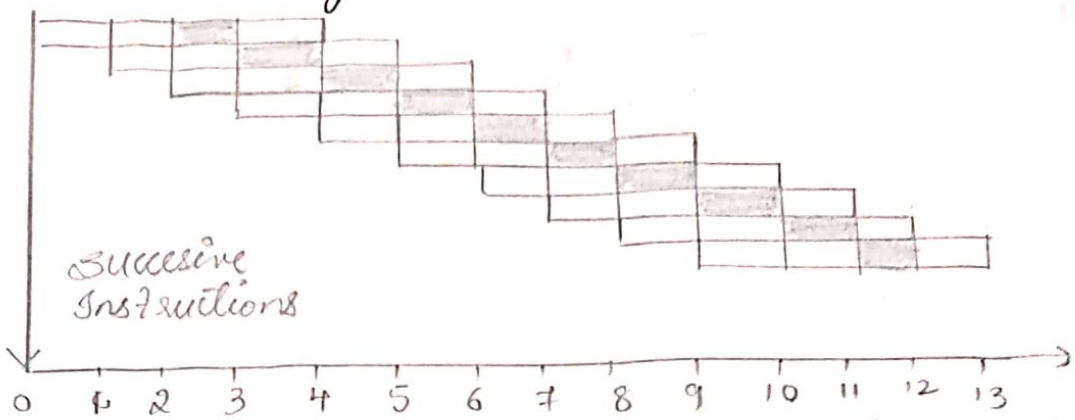


Fig (a) Scalar pipeline execution

Fig (b) Vector pipeline execution

- The pipelined execution in a vector processor is compared with that in a scalar processor in above fig.
- Fig (a) is redrawing of in which each scalar instruction executes only one operation over one data element
- Only serial issue & 1/el execution of vector instructions are shown in fig (b).
- Each vector instruction executes a string of operations, one for each element in the vector.

Symbolic Processors

- Symbolic processing has been applied in many areas, including theorem proving, pattern recognition, expert systems, knowledge engineering. Text knowledge representations, primitive operations, algorithmic behavid, memory, I/O & communications & special architectural features are different than in numerical computing.
- Symbolic processors have also been called prolog processors, Lisp processors or Symbolic manipulators
- Example: A Lisp program can be viewed as a set of functions in which data are passed from function to function.
  - The concurrent execution of these functions forms the basis for parallelism.
  - The applicative & recursive nature of Lisp requires an environment that efficiently supports stack computations & function calling.

Table: Characteristics of Symbolic Processor.

Attributes	Characteristics
Knowledge representations	Lists, relational adatabases, Scripts, Semantic nets, frames, blackboards, objects, production systems.
Common Operations	Search, Sort, pattern matching, filtering, contents, partitions, transitive closures, unification, text retrieval, set operation reasoning.
Memory Requirements	Large memory with intensive access

Attributes	Characteristics
	pattern. Addressing is often content-based. Locality of reference may not hold.
Properties of Algorithms	Non deterministic, possibly parallel & distributed computation. Data dependences may be global & irregular in pattern & granularity.
Communication patterns	Message traffic varies in size & destination, granularity & format of message units change with applications.
I/p | O/p requirements	User-guided programs, intelligent person-machine interfaces, i/p's can be graphical & audio as well as from keyboard, access to very large on-line databases
Architecture Features	Need Update of large knowledge bases, dynamic load balancing, dynamic memory allocation, H/w supported garbage collection, Stack processor architecture, Symbolic processors.

- Table Summarizes the major characteristics of Symbolic processing.
- Instead of dealing with numerical data, Symbolic processing deals with logic program, objects, scripts, blackboards, production system, Semantic n/w's, frames & ANN (Includes search, compare, logic inference, pattern matching, unification, filtering content, retrieval, set operations, transitive, closure & reasoning operations).

Example The Symbolics 3600 Lisp Processor

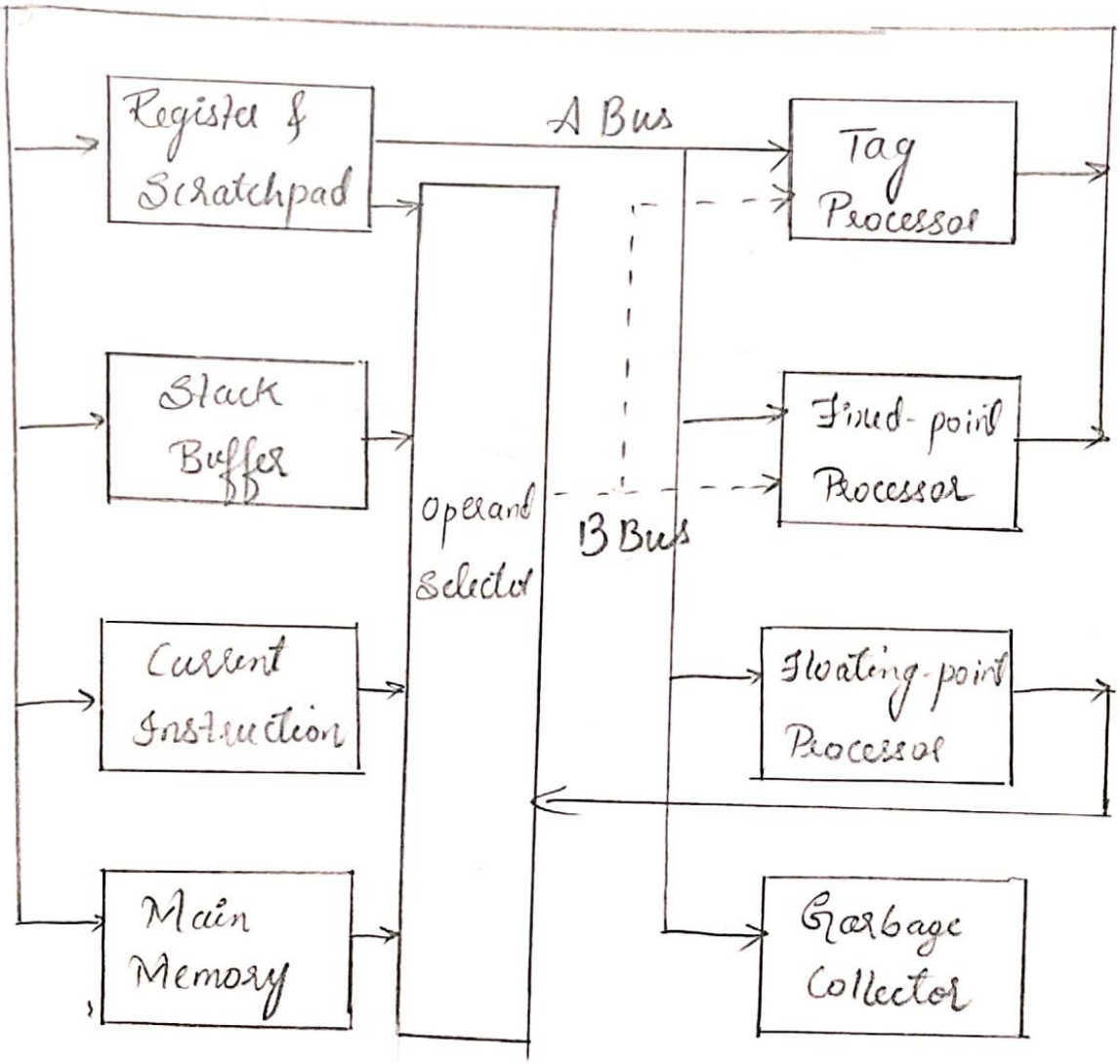


Fig: Architecture of Symbolic 3600 Lisp processor

- The processor architecture of the Symbolics 3600 is shown in fig above.
- This is a stack-oriented machine.
- The division of the overall machine architecture into layers allows the use of a pure stack model to simplify instruction-set design, while implementation is carried out with a stack-oriented machine.
- Nevertheless most operands are fetched from the stack, so the stack buffer & scratch-pad memories are

implemented as fast caches to main memory.

- The Symbolic 3600 executes most Lisp instructions in one machine cycle.
- Integer instruction fetch operands form the stack buffer & the duplicate top of the stack in the scratch-pad memory.
- Floating-point addition, garbage collection, data type checking by the tag processor, & fixed-point addition can be carried out in parallel.