RISC V ARCHITECTURE

Rajeshwari B
Department of Electronics and Communication Engineering

PES UNIVERSITY ONLINE

RISC V ARCHITECTURE

Instructions: The Language of Computer –Part-B

Rajeshwari B
Department of Electronics and Communication Engineering

MIPS Instructions

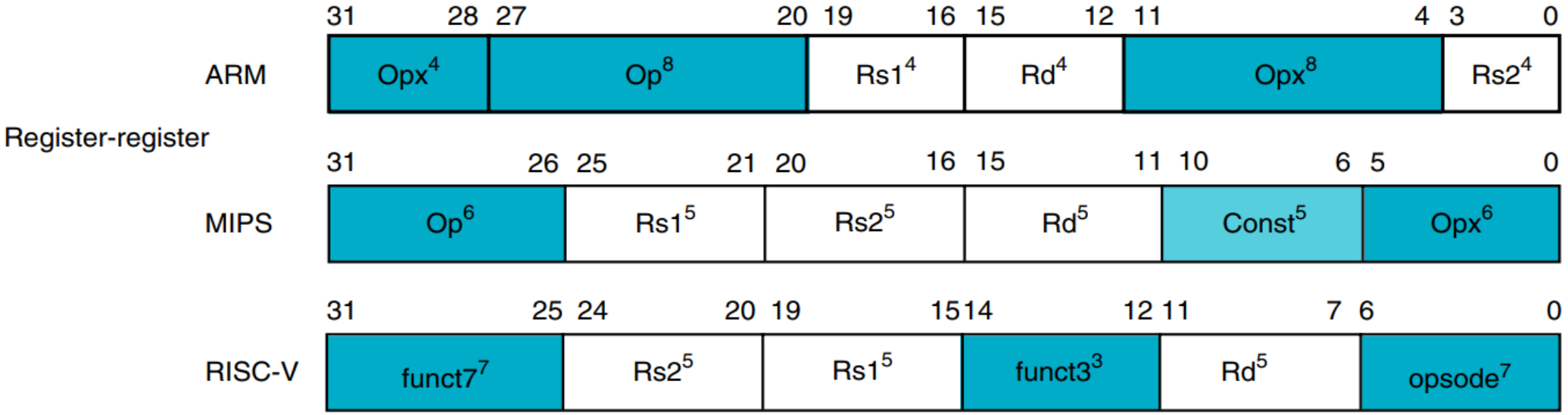
- Commercial predecessor to RISC-V
- Similar basic set of instructions
  - 32-bit instructions
  - 32 general purpose registers, register 0 is always 0
  - 32 floating-point registers
  - Memory accessed only by load/store instructions
  - Consistent use of addressing modes for all data sizes
  - There are no instructions that can load or store many registers in MIPS or RISC-V

MIPS Instructions

Different conditional branches

For <, <=, >, >= MIPS has
slt, sltu (set less than, result is 0 or 1)
Then use beq, bne to complete the branch

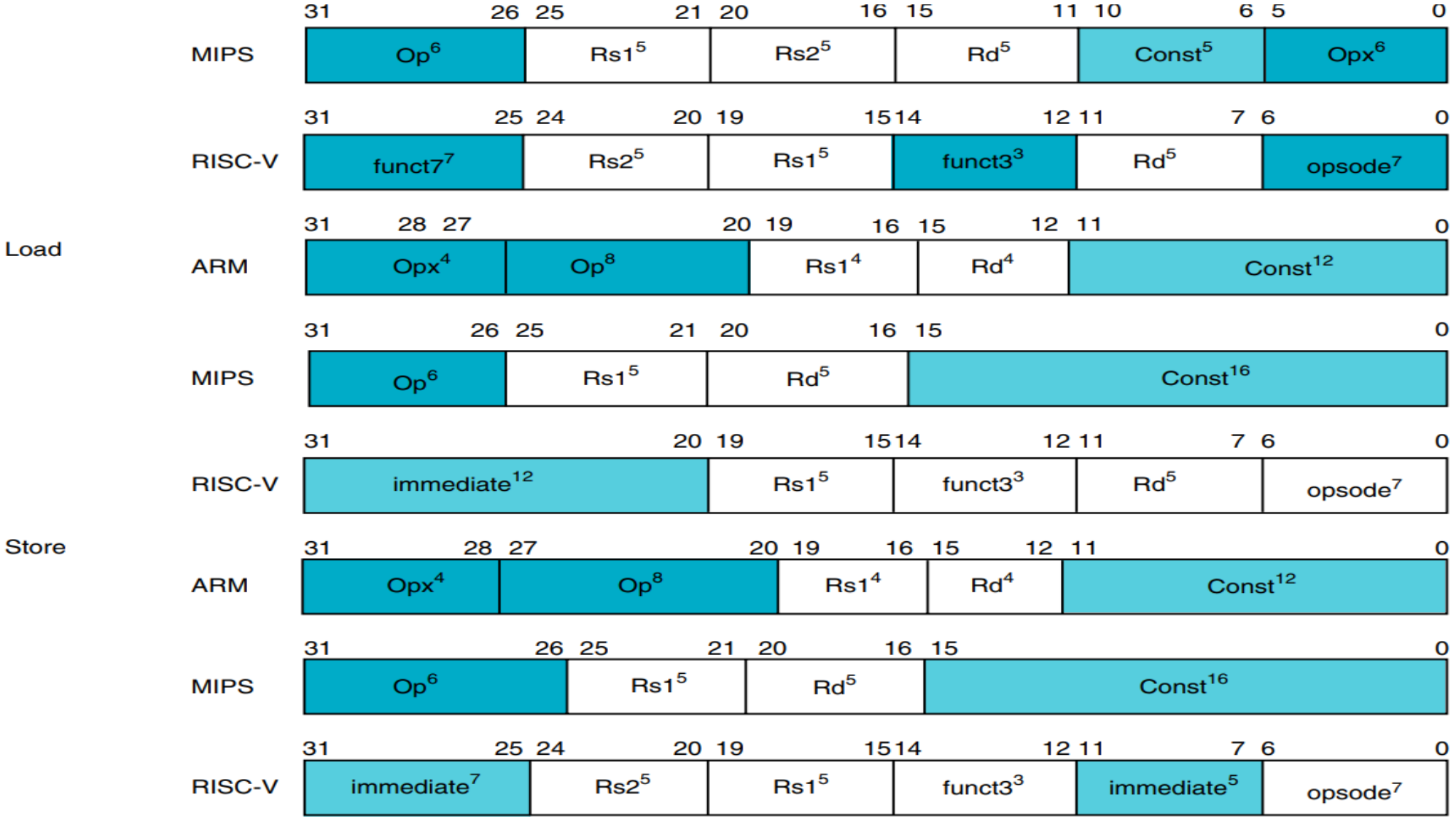
Instruction formats of ARM, RISC-V, and MIPS.



RISC V ARCHITECTURE

MIPS Instructions

Instruction formats of ARM, RISC-V, and MIPS.



Evolution with backward compatibility

- 8080 (1974): 8-bit microprocessor
  - Accumulator, plus 3 index-register pairs
- 8086 (1978): 16-bit extension to 8080
  - Complex instruction set (CISC)
- 8087 (1980): floating-point coprocessor
  - Adds FP instructions and register stack
- 80286 (1982): 24-bit addresses, MMU
  - Segmented memory mapping and protection
- 80386 (1985): 32-bit extension (now IA-32)

  Additional addressing modes and operations

  Paged memory mapping as well as segments

The Intel x86 ISA

Further evolution…

i486 (1989): pipelined, on-chip caches and FPU
- Compatible competitors: AMD, Cyrix, …

Pentium (1993): superscalar, 64-bit datapath
- Later versions added MMX (Multi-Media eXtension) instructions
- The infamous FDIV bug

Pentium Pro (1995), Pentium II (1997)
- New microarchitecture (see Colwell, *The Pentium Chronicles*)

Pentium III (1999)
- Added SSE (Streaming SIMD Extensions) and associated registers

Pentium 4 (2001)
- New microarchitecture
- Added SSE2 instructions

And further…

AMD64 (2003): extended architecture to 64 bits
EM64T – Extended Memory 64 Technology (2004)
- AMD64 adopted by Intel (with refinements)
- Added SSE3 instructions

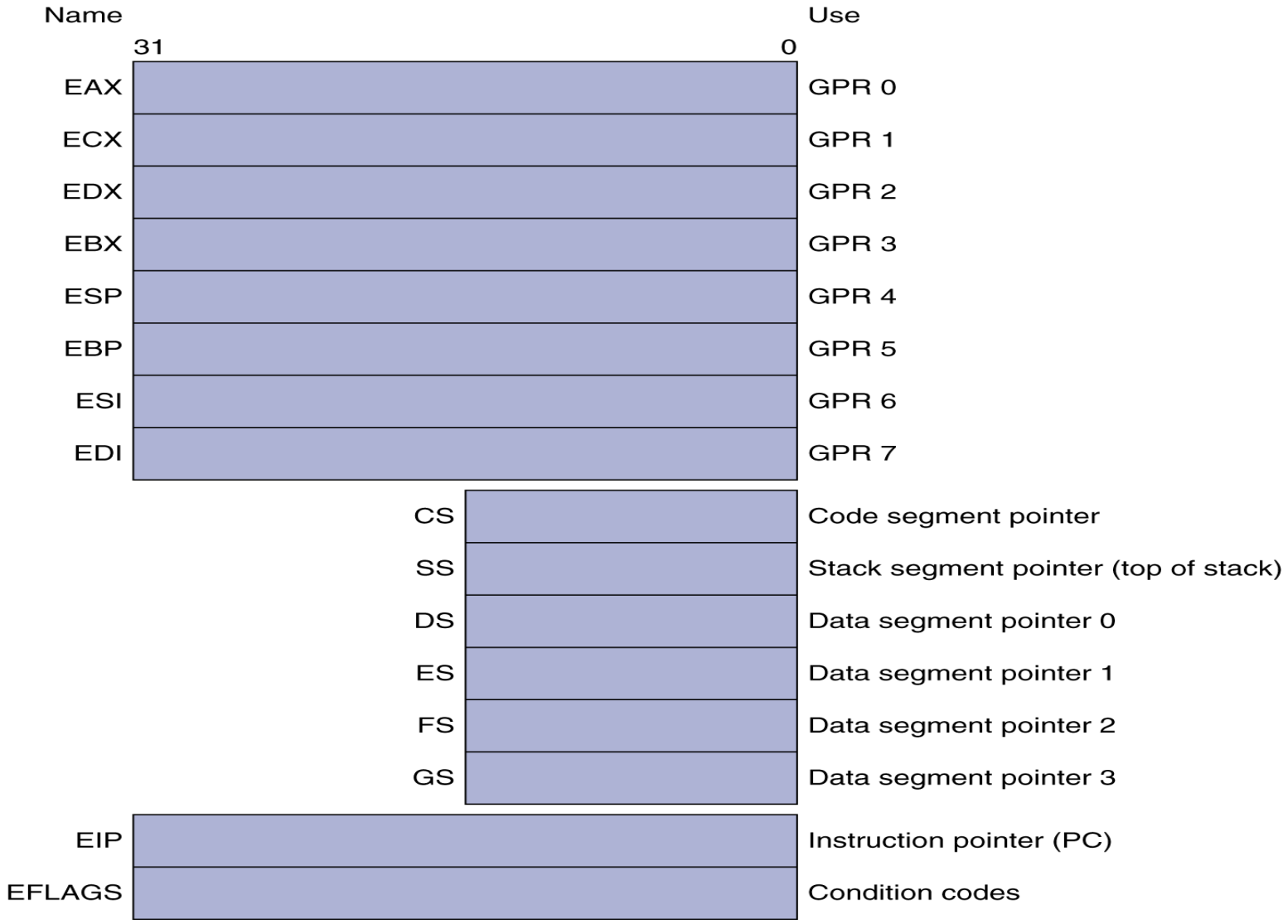
Intel Core (2006)
- Added SSE4 instructions, virtual machine support

AMD64 (announced 2007): SSE5 instructions
- Intel declined to follow, instead…

Advanced Vector Extension (announced 2008)
- Longer SSE registers, more instructions

If Intel didn't extend with compatibility, its competitors would!
- Technical elegance ≠ market success

x86 Registers and Data Addressing Modes



Basic x86 Addressing Modes

Two operands per instruction

Source/dest operand	Second source operand
Register	Register
Register	Immediate
Register	Memory
Memory	Register
Memory	Immediate

- Memory addressing modes
  - Address in register
  - Address = R_{base} + displacement
  - Address = $R_{base} + 2^{scale} \times R_{index}$ (scale = 0, 1, 2, or 3)
  - Address = $R_{base} + 2^{scale} \times R_{index}$ + displacement

Basic x86 Addressing Modes

Mode	Description	Register restrictions	RISC-V equivalent
Register indirect	Address is in a register.	Not ESP or EBP	`lw x10, 0(x11)`
Based mode with 8- or 32-bit displacement	Address is contents of base register plus displacement.	Not ESP	`lw x10, 40(x11)`

Base plus scaled index	The address is Base + (2^{Scale} × Index) where Scale has the value 0, 1, 2, or 3.	Base: any GPR
		Index: not ESP
			`slli x12, x12, 2`
			`add x11, x11, x12`
			`lw x10, 0(x11)`

Base plus scaled index with 8- or 32-bit displacement	The address is Base + (2^{Scale} × Index) + Displacement where Scale has the value 0, 1, 2, or 3.	Base: any GPR
		Index: not ESP
			`slli x12, x12, 2`
			`add x11, x11, x12`
			`lw x10, 40(x11)`

The Intel x86 ISA

The x86 integer operations can be divided into four major classes

1. Data movement instructions, including move, push, and pop.

2. Arithmetic and logic instructions, including test, integer, and decimal arithmetic operations.

3. Control flow, including conditional branches, unconditional branches, calls, and returns.

4. String instructions, including string move and string compare.

The Intel x86 ISA

Conditional branches: on the x86 are based on condition codes or flags.

Condition codes are set as a side effect of an operation; most are used to compare the value of a result to 0.

Branches then test the condition codes.
PC-relative branch addresses must be specified in the number of bytes

Instruction	Meaning
Control	**Conditional and unconditional branches**
`jnz, jz`	Jump if condition to EIP + 8-bit offset; `JNE` (for `JNZ`), `JE` (for `JZ`) are alternative names
`jmp`	Unconditional jump—8-bit or 16-bit offset
`call`	Subroutine call—16-bit offset; return address pushed onto stack
`ret`	Pops return address from stack and jumps to it
`loop`	Loop branch—decrement ECX; jump to EIP + 8-bit displacement if ECX≠0

The Intel x86 ISA

Some typical x86 instructions and their functions.

Instruction	Function
je name	if equal(condition code) {EIP=name}; EIP-128 <= name < EIP+128
jmp name	EIP=name
call name	SP=SP-4; M[SP]=EIP+5; EIP=name;
movw EBX,[EDI+45]	EBX=M[EDI+45]
push ESI	SP=SP-4; M[SP]=ESI
pop EDI	EDI=M[SP]; SP=SP+4
add EAX,#6765	EAX= EAX+6765
test EDX,#42	Set condition code (flags) with EDX and 42
movsl	M[EDI]=M[ESI]; EDI=EDI+4; ESI=ESI+4

RISC V ARCHITECTURE

The Intel x86 ISA

Data transfer	Move data between registers or between register and memory
move	Move between two registers or between register and memory
push, pop	Push source operand on stack; pop operand from stack top to a register
les	Load ES and one of the GPRs from memory
Arithmetic, logical	**Arithmetic and logical operations using the data registers and memory**
add, sub	Add source to destination; subtract source from destination; register-memory format
cmp	Compare source and destination; register-memory format
shl, shr, rcr	Shift left; shift logical right; rotate right with carry condition code as fill
cbw	Convert byte in eight rightmost bits of EAX to 16-bit word in right of EAX
test	Logical AND of source and destination sets condition codes
inc, dec	Increment destination, decrement destination
or, xor	Logical OR; exclusive OR; register-memory format
String	**Move between string operands; length given by a repeat prefix**
movs	Copies from string source to destination by incrementing ESI and EDI; may be repeated
lods	Loads a byte, word, or doubleword of a string into the EAX register

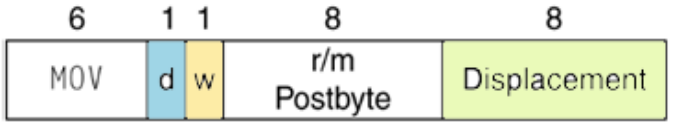
The Intel x86 ISA

x86 Instruction Encoding

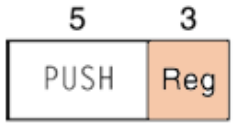
a. JE EIP + displacement

b. CALL

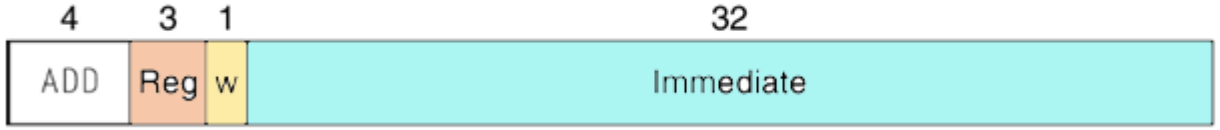
c. MOV EBX, [EDI + 45]



d. PUSH ESI



e. ADD EAX, #6765



f. TEST EDX, #42

Variable length encoding

Overall, instructions may vary from 1 to 15 bytes in length.

The long length comes from extra 1-byte prefixes, having both a 4-byte immediate and a 4-byte displacement address.

Postfix bytes specify addressing mode Prefix bytes modify operation

Many instructions contain the 1-bit field w, which says whether the operation is a byte or a doubleword.

The d field in MOV is used in instructions that may move to or from memory and shows the direction of the move.
The ADD instruction requires 32 bits for the immediate field, because in 32-bit mode
The immediate field in the TEST is 32 bits long because there is no 8-bit mode immediate for test in 32-bit mode.

x86 Conclusion

Though the x86 is more difficult to build than computers like RISC-V and MIPS, but the large market meant in the PC era that AMD and Intel could afford more resources to help overcome the added complexity.

In the post-PC era, however, despite considerable architectural and manufacturing expertise, x86 has not yet been competitive in the personal mobile device.

THANK YOU

Rajeshwari B
Department of Electronics and Communication Engineering
rajeshwari@pes.edu