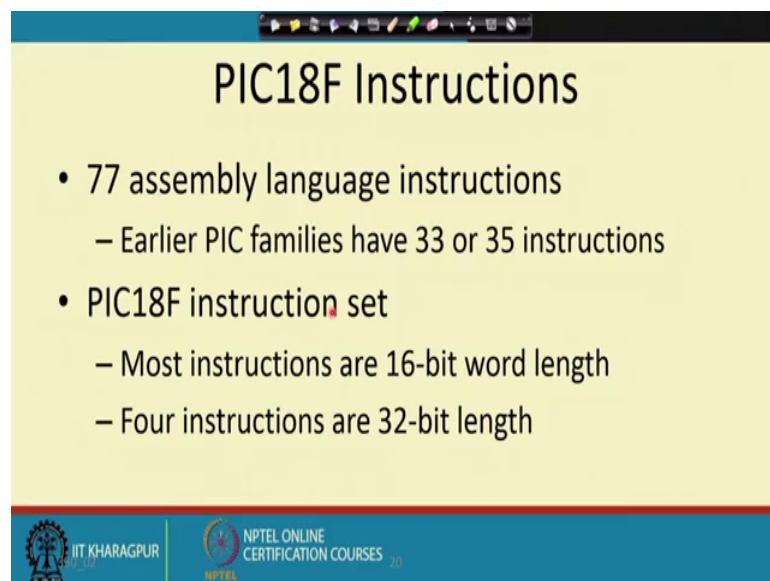


**Microprocessors and Microcontrollers**  
**Prof. Santanu Chattopadhyay**  
**Department of E & EC Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 50**  
**PIC, AVR**

So, the PIC 8 it 18F instructions; So, as I was telling that there are 77 assembly language instructions in PIC.

(Refer Slide Time: 00:17)



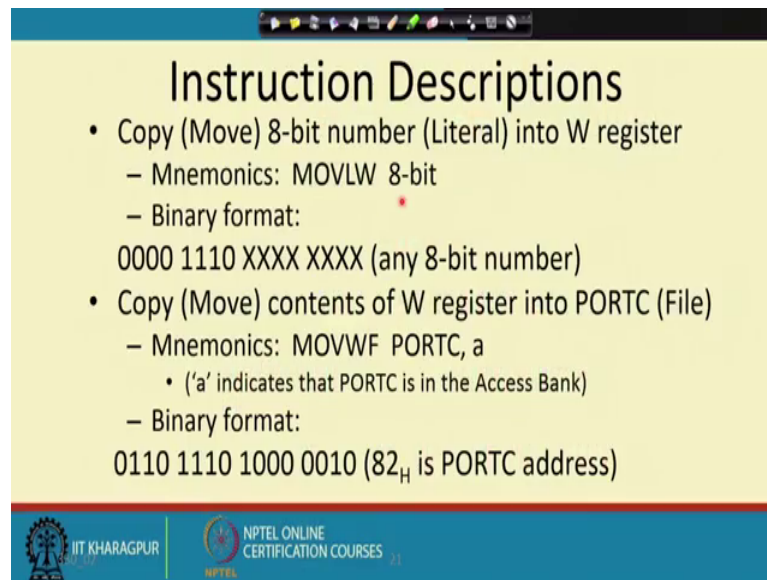
**PIC18F Instructions**

- 77 assembly language instructions
  - Earlier PIC families have 33 or 35 instructions
- PIC18F instruction set
  - Most instructions are 16-bit word length
  - Four instructions are 32-bit length

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Earlier PIC versions they are 30 to 35 instructions and then this instruction set most of the instructions are 16-bit word length and there are 4 instructions that have 32 bit long. So, that way those instructions are more powerful, but otherwise the instructions are uniform. So, they are 16-bit wide, but only 4 instructions they are 32 bit wide.

(Refer Slide Time: 00:49)



The slide is titled "Instruction Descriptions" and contains two bullet points. The first bullet point describes the "Copy (Move) 8-bit number (Literal) into W register" instruction, listing its mnemonics as "MOVLW 8-bit" and its binary format as "0000 1110 XXXX XXXX (any 8-bit number)". The second bullet point describes the "Copy (Move) contents of W register into PORTC (File)" instruction, listing its mnemonics as "MOVWF PORTC, a" and its binary format as "0110 1110 1000 0010 (82<sub>H</sub> is PORTC address)". The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES at the bottom.

### Instruction Descriptions

- Copy (Move) 8-bit number (Literal) into W register
  - Mnemonics: MOVLW 8-bit
  - Binary format:  
0000 1110 XXXX XXXX (any 8-bit number)
- Copy (Move) contents of W register into PORTC (File)
  - Mnemonics: MOVWF PORTC, a
    - ('a' indicates that PORTC is in the Access Bank)
  - Binary format:  
0110 1110 1000 0010 (82<sub>H</sub> is PORTC address)


If you look into these instructions then the first category of instruction is the copy instructions or move instruction they have. So, this is a move copy 8-bit number into W register. So, the format is like MOVLW 8-bit. So, so, move literal so, it is like move literal to W register 8-bit number. So, this is some sort of move for immediate. So, the coding will be like this that 0000 1110 and then this is the 8-bit number that we have, ok.

Then another interesting instruction may be that copy contents of W register into PORTC. So, the this is also for moving to some moving the content of this W register onto some memory location. So, MOVWF, ok so, this is working register PORTC comma a. So, this it will go to PORTC and here the a will indicate that the PORTC is in the current access bank. So, that is the current bank selection whatever is there. So, it will be there it will be used for that and 82 H is the PORTC address. So, this is the 82 H. So, it will be useful there and rest of the bits are fixed for this operation.

(Refer Slide Time: 02:05)

## Illustrative Program

- Problem statement:
  - Write instructions to light up alternate LEDs at PORTC
- Hardware:
  - PORTC
    - Bidirectional (input or output) port
    - Setup as output port for display
  - Logic 1 will turn on an LED



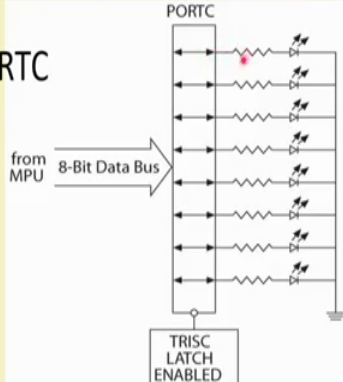
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

We will try to see one very simple program. So, we will try to write a few instructions to light up alternate LEDs at PORTC, ok. So, in the hardware connection so, we will have this PORTC and is a bidirectional port input, and it can be configured as input or output. So, you have to set it for output mode for display and then also logic one will turn on an LED. So, that is the structure. So, for this we will try to write the program.

(Refer Slide Time: 02:39)

## Illustration

- Interfacing LEDs to PORTC



IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, this is the structure. So, from the 8-bit data bus it will come to PORTC and we have got these registers register and then register and then the LED. So, if you put a one here

so, the corresponding LED will glow and there is a tri state latch enable signal the TRISC is a special register. So, you if you want to enable PORTC so, you have to enable this latch then only the values will be going to are actually this PORTC has to be configured as either input or output. So, for that purpose this TRISC so, this has to be controlled.

(Refer Slide Time: 03:13)

**Illustration**

- Program (software)
  - Logic 0 to TRISC sets up PORTC as an output port
  - Byte 55<sub>H</sub> turns on alternate LEDs

• MOV LW	00	;Load W register with 0
• MOV WF	TRISC	;Set up PORTC as output
• MOV LW	0x55	;Byte 55 <sub>H</sub> to turn on LEDs
• MOV WF	PORTC	;Turn on LEDs
• SLEEP		;Power down

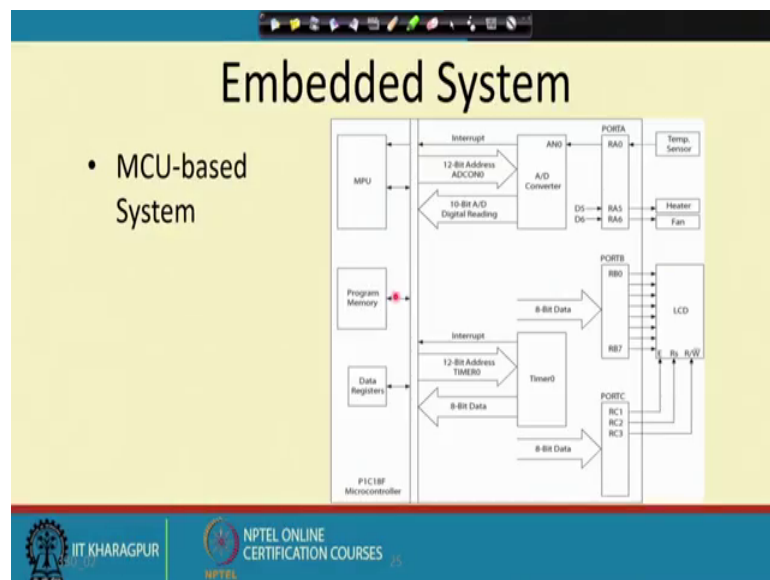
So, for the software side first we write logic 0 to TRISC register for configuring PORTC as an output port, and then we will move 55 H to alternate LEDs. So, how do we do this? So, MOV LW 00, So, W register will be loaded with the immediate word 0, and then MOV WF try TRISC, so, this will set PORTC as output port, and now, after that I can put the byte 55 H on to PORTC. So, just like 8051; 8051 for output we did not need to configure it, but whenever we are trying to con configure it as input so, you have to write a 0 first onto the write a one to the port first. So, that the latch was getting disabled.

So, here also similar thing that PORTC the PORTC is a bi directional port as we see here so, for programming it is an output mode. So, you should have this TRISC latch set to set so that we can get this we this latch is we configure this PORTC in the output mode. So, if you send a 0 to TRISC then it will set up the output PORTC as output mode if you send it as all one then it will be configured it has input mode. Then we are outputting this 55 x, so, that is for the alternating LEDs will be on alternate LEDs will be on and then this pattern so, this is moved on to the W register.

So, you cannot put this pattern directly on to PORTC. So, you have to put it on the W register and then MOVWF PORTC. So, this will turn on the LEDs because this 55 x will go to PORTC and it will be turned on and then it will go to this a sleep. So, that is the power down mode. So, it will so, it is assumed that it will be after turning it on so, it is going to power down mode.

So, if you want that they should blink then you should put a delay here and then do a shifting of this W register from 55 x it should come to the status or a x and that way you can put a loop around this. So, that you can get alternate blinking of LEDs, that is not shown in this program.

(Refer Slide Time: 05:34)



So, for an embedded application you see that this is going to be useful because we have got this microprocessor unit program memory and data registers. So, that are there, then we have got these AD converters are there. So, AD converter we has got analog 0 channel. So, this is a typical example like we have connected one temperature, sensor, one heater fan and one LCD panel, ok. So, it may be connected like this that this we are this temperature sensor will sense one analog value, ok.

So, that is this that is connected to the AD converter and for that connection we need to use this RA0 pin of this PORTA. So, PORTA, I am connecting it in the input mode and this temperature sensor is connected to RA0 which is a multiplexed operation RA0 engine is a multiplexed operation.

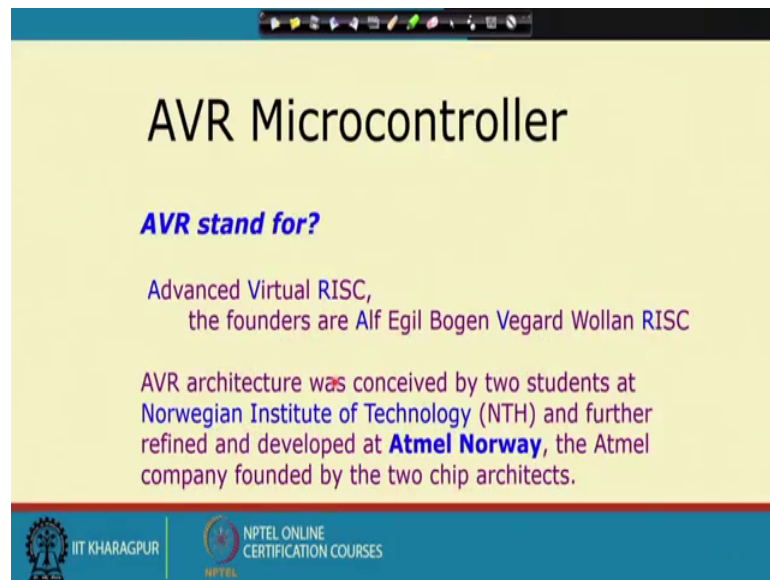
So, it gets it here. So, whenever this a d converter finishes converting one temperature value. So, this the voltage value that is coming from the sensor into the digital value it will send an interrupt to the microprocessor. So, there with M P U unit and when it is we get after getting that interrupts. So, it will be it will be it will be reading this 15 bit value of the AD converter into this processor and then after depending upon the temperature we may like to turn on the heater or turn on the fan.

If the temperature is above the setting the set value then the fan should be turned on and if it is below the set value the heater should be turned on. So, that can be controlled by these two bits RA5 and RA6 of this port and accordingly the bits D5 and D6 can be controlled of by the microprocessor for PORTA. PORTB and PORTC, so, they are used for this LCD interface, in a PORTB we send the pattern that we want to display and this the other controls the is the R s and read writes these are the controls for the LCD. So, they are coming from this r c lines this PORTC RC 1, RC 2 and RC 3 bits. So, they will be controlling the LCD panel.

So, this way you can develop a very simple system using this PIC microcontrollers or for that matter it can be any other microcontroller, but we can design embedded systems built around these microcontrollers and sensors ok. If so; the advantage is that ADC timer and this LCD. So, the sorry ADC timer and this digital I also they are all part of the microcontroller that helps in the design of the system. So, after this we look into another microcontroller which is known as the which is known as the AVR microcontroller.

So, this is also quite common and this is used very much in many of these processors many of this embedded system designs because of the same reason because it is one of the powerful of microcontrollers that we have. So, and again there is a range of such microcontrollers, in the AVR series also you can find a range of such microcontrollers.

(Refer Slide Time: 08:57)



**AVR Microcontroller**

**AVR stand for?**

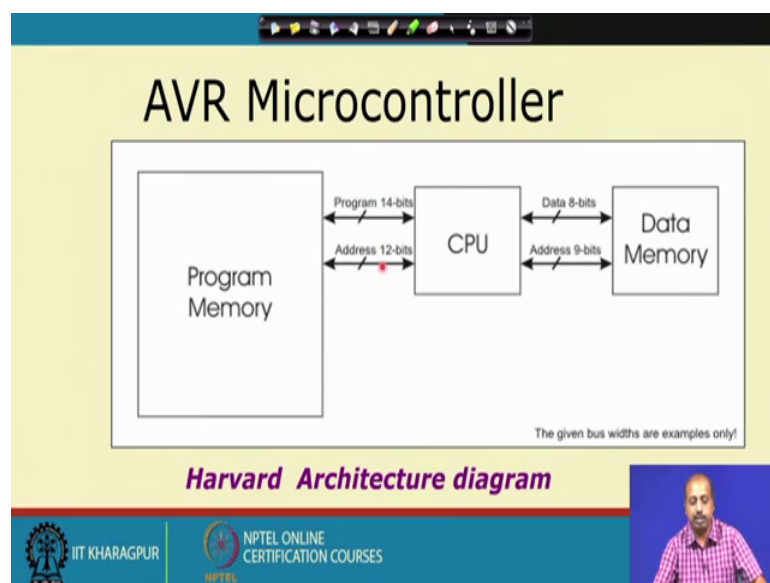
Advanced Virtual RISC,  
the founders are Alf Egil Bogen Vegard Wollan RISC

AVR architecture was conceived by two students at Norwegian Institute of Technology (NTH) and further refined and developed at **Atmel Norway**, the Atmel company founded by the two chip architects.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, AVR it stands for Advanced Virtual RISC, ok. So, this is founders are Alf Egil Bogen and Vegard Wollan RISC. So, now, they are as they were actually to university student. So, when they when the student at NTH they started the thinking about this architecture and later on they developed the structure and refined it into when to acquire and develop the company Atmel Norway and then the Atmel company was founded by these two chip architects. So, this is one of the success stories of the university projects going into the product.

(Refer Slide Time: 09:39)



**AVR Microcontroller**

**Harvard Architecture diagram**

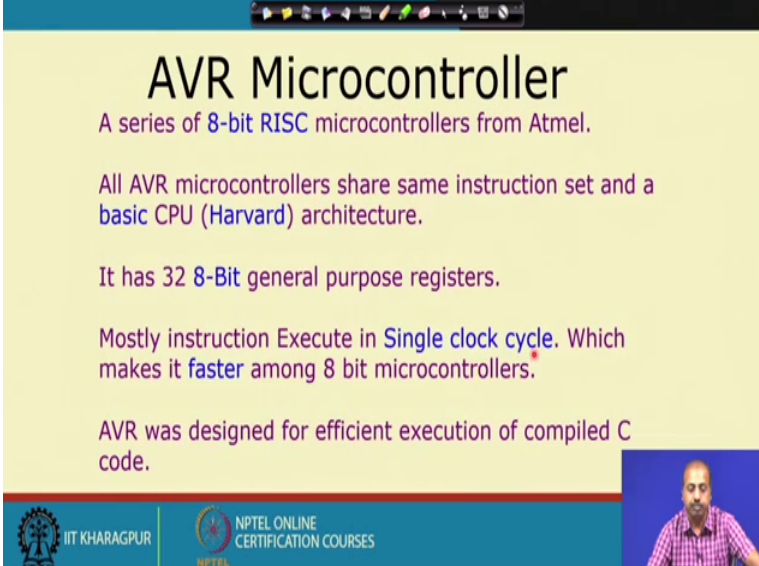
The diagram illustrates the Harvard Architecture of the AVR Microcontroller. It shows three main components: Program Memory, CPU, and Data Memory. The CPU is connected to Program Memory via a 14-bit Program bus and a 12-bit Address bus. The CPU is also connected to Data Memory via an 8-bit Data bus and a 9-bit Address bus. A note at the bottom right of the diagram states: "The given bus widths are examples only!".

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, as far as the microcontroller is concerned so, it follows Harvard Architecture again. So, this is program memory. So, there is a CPU. So, it is connected to you can have program memory and data memory and this program memory and data memory they will have their individual buses, the program memory address bus and data bus and flow data memory program is bus and data bus. So, this width values 14 bits 12 bits.

So, they are they are varying like if you look into different series of AVR microcontrollers then these values will be vary accordingly we can have different structures.

(Refer Slide Time: 10:15)



The slide features a yellow background with a blue header and footer. The title 'AVR Microcontroller' is in large black font. Below it, a purple subtitle reads 'A series of 8-bit RISC microcontrollers from Atmel.' The main content consists of five purple bullet points: 'All AVR microcontrollers share same instruction set and a basic CPU (Harvard) architecture.', 'It has 32 8-Bit general purpose registers.', 'Mostly instruction Execute in Single clock cycle. Which makes it faster among 8 bit microcontrollers.', and 'AVR was designed for efficient execution of compiled C code.' The footer contains the IIT Kharagpur logo on the left and the NPTEL Online Certification Courses logo on the right. A small video inset of a man in a pink shirt is visible in the bottom right corner.

**AVR Microcontroller**  
A series of 8-bit RISC microcontrollers from Atmel.

- All AVR microcontrollers share same instruction set and a basic CPU (Harvard) architecture.
- It has 32 8-Bit general purpose registers.
- Mostly instruction Execute in Single clock cycle. Which makes it faster among 8 bit microcontrollers.
- AVR was designed for efficient execution of compiled C code.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, these AVR microcontrollers they are a series of 8-bit RISC microcontrollers from Atmel. So, unlike arm which is a 32 bit RISC architecture. So, this AVR is an 8-bit RISC architecture. So, this is. So, naturally the structure is going to be much simpler like all the operations if they are 8-bit operation then naturally the hardware module hardware operate it will be less and possibly it will be able to operate at a higher speed. So, they share the same instruction set as the basic CPU. So, all these AVR series of microcontrollers they will be using the same set of instructions for as we had in the basic AVR microcontroller and they follow the Harvard architecture 32 8-bit general purpose registers.

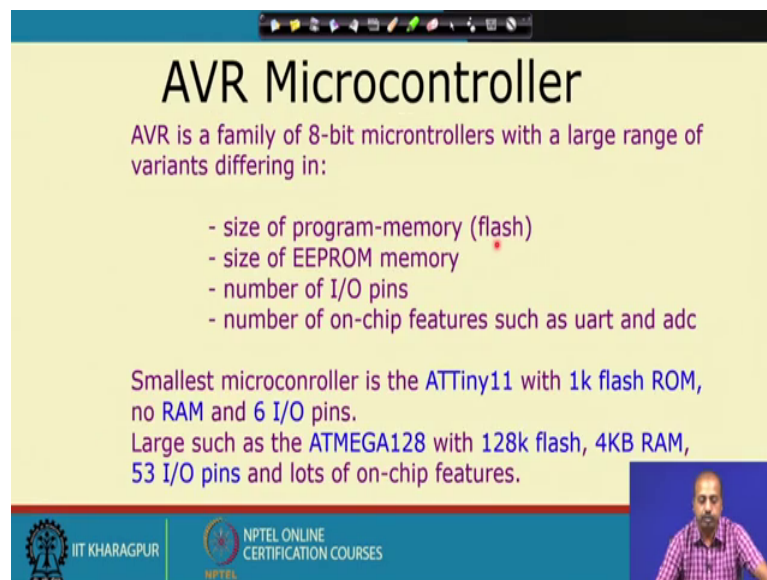
So, it has 32 8-bit general purpose registers. So, that is a large number of registers you see. So, so, that is that makes it suitable for RISC architecture and most instructions



execute in a single clock cycle. So, which makes it faster among 8-bit microcontrollers. So, in a single clock cycle the entire instruction is executed. So, there will be pipelining and parallelization whatever is there inside, but ultimately in a single clock cycle the entire instruction will be executed. So, architecture is very much optimized towards that. Mostly instructions execute a single clock cycle which may say. So, that makes it better and this is efficient execution of compiled C code.

So, this is another very interesting thing that these AVR designers they did not expect the users to be very proficient in AVR assembly language programming. So, it was taken that from there it is you can write your program in C code and then there will be efficient compiler which will compile this C code into AVR code and that philosophy is followed by many other companies particularly if you look into say the a digital signal processors. So, there also you will find that the programs are it mean high level language only and then through this efficient compilation so, they are translated into the machine language. So, it is no more like hand written program. So, they are all automated programs generated from high level language.

(Refer Slide Time: 12:41)



**AVR Microcontroller**

AVR is a family of 8-bit microcontrollers with a large range of variants differing in:

- size of program-memory (flash)
- size of EEPROM memory
- number of I/O pins
- number of on-chip features such as uart and adc

Smallest microcontroller is the ATTiny11 with 1k flash ROM, no RAM and 6 I/O pins.  
Large such as the ATMEGA128 with 128k flash, 4KB RAM, 53 I/O pins and lots of on-chip features.

The slide includes a navigation bar at the top, the IIT Kharagpur logo, and the NPTEL Online Certification Courses logo. A small video inset in the bottom right corner shows a man speaking.

So, AVR is a family of 8-bit microcontrollers with large range of variants differing in size of the program memory size of the EEPROM memory number of I O pins then number of on chip features such as user tend a either the universal a synchronous receiver transmitter and the adc. Smallest of this series is a tie ATTiny11, ok. So, that is

that has got one kilo flash ROM there is no RAM and there are 6 I O pins. So, this is a simplest possible processor that we have and the large one such as ATMEGA128 it has got 128 kilo flash 4 kilo byte over RAM 53 I O pins and lots of on chip features like adc user and all that. So, there many such features that are on chip. So, we will try to see a broad overview of all these processors.

(Refer Slide Time: 13:37)

Part Number	Pins	Flash	EEPROM	RAM
90S1200	20	1K	64 Bytes	0
90S2313	20	2K	128	128
90S2323	8	2K	128	128
90S2333	28	2K	128	128
90S4433	28	4K	256	128
90S4414	40	4K	256	256
90S8515	40	8K	512	512
90S4434	40	4K	256	256
90S2343	8	2K	128	128
Mega103	64	128K	4096	4096
Mega603	64	64K	2048	4096
Tiny10	8	1K	64	0
Tiny12	8	1K	64	0
Tiny13	8	2K	128	128

So, if you look into say this 90S1200 it is a 20 pin chip with a flash of 1 kilo and EEPROM of 64 bytes and there is no RAM. Then 90S2313, so, it has got other features as EEPROM ideas doubled and there is a 128 RAM locations are introduced. Coming to this me mega series, so, Mega603 or Mega103 it has got a 64 pin is a 64 pin chip 128 kilo of flash space 4096 bytes of EEPROM. So, 4 kilo EEPROM and then we have got 4 0 as a 4 kilo of RAM space. So, that is one of the very advanced one.

So, this way we can have a series of atmega processors or after these AVR processors and they are capacities will be vary depending upon the series that we use.

(Refer Slide Time: 14:32)

**AVR AT90S2313 Microcontrollers**

This is a microcontroller of AVR series from Atmel.

**High-performance and Low-power RISC Architecture**

It is a low voltage (2.7V - 6V), high performance CMOS 8-bit micro controller based on the AVR **RISC architecture**

Since it is a microcontroller from AVR series ,it is also using Harvard Architecture

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

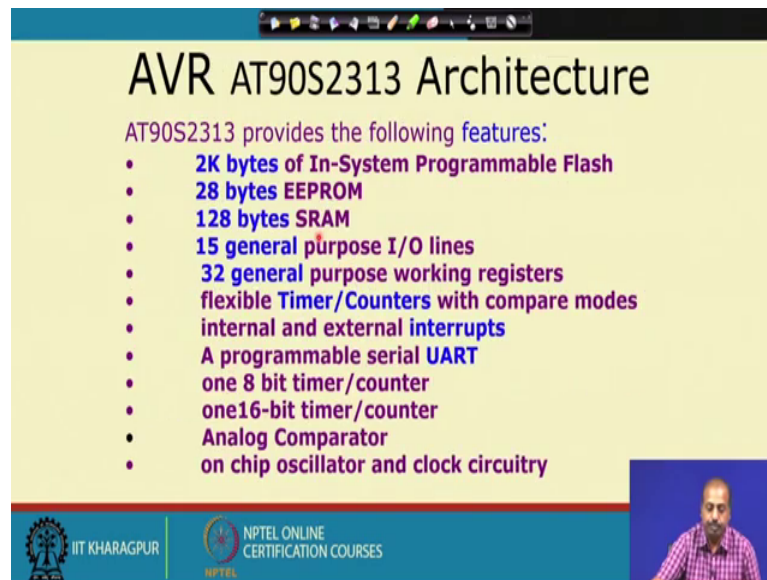
The slide features a yellow background with a blue header and footer. A small video inset of a man in a purple shirt is visible in the bottom right corner.

So, we will look into once some such representative microcontrollers. So, the this discussion is for AT90S2313 series of microcontrollers. So, this is a microcontroller of AVR series, ok. So, it is a high performance low power RISC architecture it is a low voltage. So, the operating voltage can be 2.7 volt to 6 volts.

So, this is another very good thing. So, you can have a wide range of supply voltage on which the processor will work maybe when you are operating at 2.7 volt you do not get the same performance as when you operate at the 6 volt, but this power and performance the trade off can be there; like you can you may want to you may be satisfied with a lower degree of performance, but the power consumption should be low.

So, in that case you can operate it as a operate it as a lower voltage and the other or the other side so, you may want that performance should be good and I do not matter mind giving some more power. So, then you can operate at a higher voltage. The high performance CMOS 8-bit microcontroller; So, it has it has built on CMOS technology and it is based on the AVR RISC architecture since a microcontroller for AVR series. So, it is also using Harvard architecture, ok. So, that is the same.

(Refer Slide Time: 15:50)



**AVR AT90S2313 Architecture**

AT90S2313 provides the following features:

- **2K bytes of In-System Programmable Flash**
- **28 bytes EEPROM**
- **128 bytes SRAM**
- **15 general purpose I/O lines**
- **32 general purpose working registers**
- **flexible Timer/Counters with compare modes**
- **internal and external interrupts**
- **A programmable serial UART**
- **one 8 bit timer/counter**
- **one 16-bit timer/counter**
- **Analog Comparator**
- **on chip oscillator and clock circuitry**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

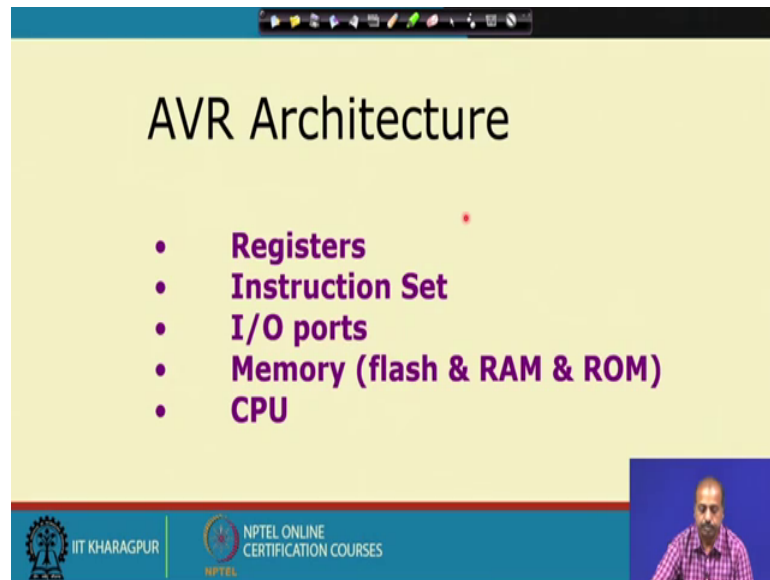
So, features; so, it has got 2 kilo byte of in system programmable flash that where you can have this program memory then there is 28 byte of EEPROM. So, you can have some EEPROM where you can have only 28 bytes are there, but the essential values can be programmed there.

Then we have got one 28 bytes of SRAM that is for the scratchpad purpose 15 general purpose I O line basically, that I O ports then 32 general purpose working registers. So, large number of registers we have got timer counters with flexible timer counter will compare modes. So, we can we can compare the value of these timers and counters with some preset values. Then internal and external interrupts the interrupts are their programmable serial universal asynchronous receiver transmitter the UART, then we have got one 8-bit timer counter then we can have one 16-bit timer counter. So, one 8-bit timer counter, one 16-bit timer counter so, you have got analog comparator. So, this is another very important thing.

So, many times we need to oh have two analog signals and we need to compare between them. So, in normal proc processing that we have seen so far. So, what you have to do is we have to take those we have to convert those analog values into digital and by means of some analog to digital converter and then using that converter we have to use the we have to run the program. So, you have to get compare at the digital level, but if we can do it at the analog level itself then this additional a d converters are not necessary. So,

that is there. So, we can have this analog comparator in the AVR architecture. Then there is on chip oscillator and clock circuitry for clock generation. So, you have got on chip oscillator. So, if you there so, you do not need the crystals and all that. So, you can the oscillator will be able to generate the clock.

(Refer Slide Time: 17:51)



The slide is titled "AVR Architecture" and lists five key components: Registers, Instruction Set, I/O ports, Memory (flash & RAM & ROM), and CPU. It includes logos for IIT Kharagpur and NPTEL Online Certification Courses, and a small video inset of the presenter.

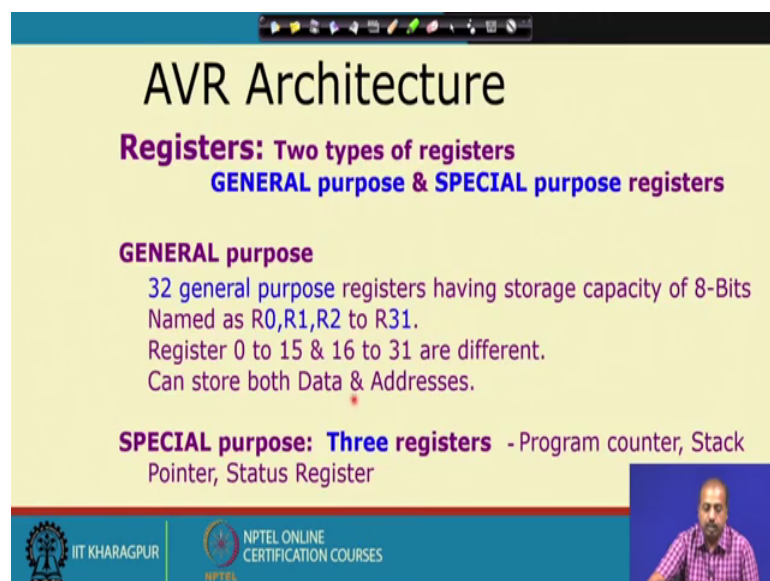
## AVR Architecture

- **Registers**
- **Instruction Set**
- **I/O ports**
- **Memory (flash & RAM & ROM)**
- **CPU**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, will be looking into this AVR architectures particularly we look into the registers the instruction set I O ports, memory and the CPU structure.

(Refer Slide Time: 18:02)



This slide details the register structure of AVR architecture, distinguishing between general purpose and special purpose registers. It includes logos for IIT Kharagpur and NPTEL Online Certification Courses, and a small video inset of the presenter.

## AVR Architecture

**Registers: Two types of registers**  
**GENERAL purpose & SPECIAL purpose registers**

**GENERAL purpose**  
32 general purpose registers having storage capacity of 8-Bits  
Named as R0,R1,R2 to R31.  
Register 0 to 15 & 16 to 31 are different.  
Can store both Data & Addresses.

**SPECIAL purpose: Three registers** - Program counter, Stack Pointer, Status Register

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, registers; there are two types of registers one category is called general purpose or that is called special purpose. So, general purpose register there are 32 general purpose registers having storage capacity of 8-bits. So, named R0, R1 up to R 31 so, these R 32 general purpose registers and out of that register R0 to R15 and R16 to R31, they are different.

So, they are as if they can they are two different banks and some something like that. Can store both data and addresses, unlike other category well all the registers are not equally capable. Like in 8051 you see that we have got the registers, but for indirect addressing we can only use R0 and R1 or if you look into the 8085 then we have got this HL pair that can be used as memory pointer, but not all of them.

So, you see that these general purpose registers in case of AVR architecture. So, they can be now, all these registers can be used as data or address. And, there are three special purpose registers; program counter, stack pointer and status register. So, these are the special purpose registers that we have in the AVR architecture.

(Refer Slide Time: 19:17)

**AVR Architecture**  
**Pointer Register**  
 Three 16-bit address registers pairs of registers 26 to 31 have extra meaning in AVR assembly.  
 X (r27:r26), y (r29:r28), z (r31:r30).

pointer	Sequence
X	• Read/Write from address X, don't change the pointer

X-register: 15-bit register with bits 7-15 and 0-7, labeled R27 (S16) and R26 (S1A).

Y-register: 15-bit register with bits 7-15 and 0-7, labeled R29 (S1C) and R28 (S1B).

Z-register: 15-bit register with bits 7-15 and 0-7, labeled R31 (S1E) and R30 (S1D).

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

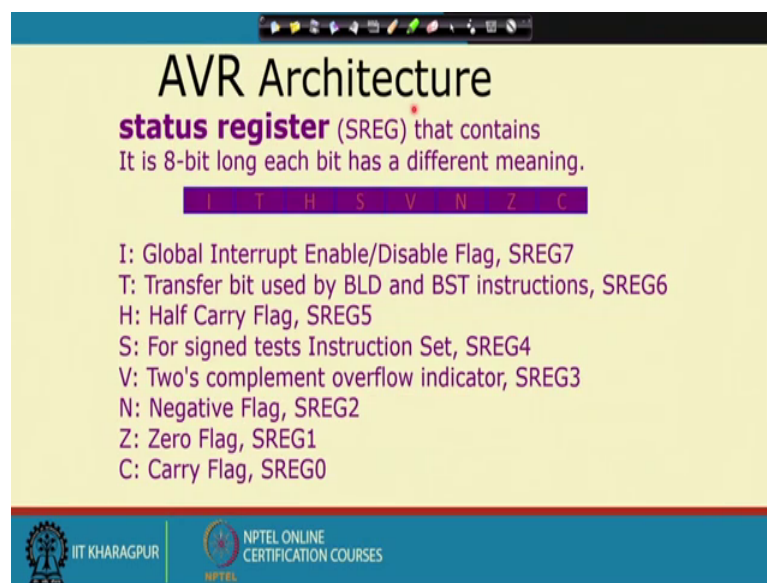
So, pointer register; so, there are three additional the 16-bit register pairs registers 26 to 31. So, they are called the pointer register. So, are so, they are called X, y and z pointers. So, pointer X is this one so, this is consisting of register number 26 and 27. So, they are this gives me the 16-bit address and this is they are called this is the X-pointer. Similarly,



we have got y-pointer which is another 16-bit consisting of the registers r-register 28 and 29 and z-register consisting of the registers 30 and 31.

So, read write from address X and they do not change the pointer. So, if you if you. So, if you use this X pointer in your instruction then it will be using this pair as the memory location address from where the value should be read. So, we have got three such registers, X, y and z. So, if you have got pointer supported in high level language to convert it into assembly code this pointer registers can be used.

(Refer Slide Time: 20:27)



The slide is titled "AVR Architecture" and describes the "status register (SREG)". It states that the register is 8-bit long and each bit has a different meaning. A purple bar contains the letters I, T, H, S, V, N, Z, C. Below this, a list explains each bit: I (Global Interrupt Enable/Disable Flag, SREG7), T (Transfer bit used by BLD and BST instructions, SREG6), H (Half Carry Flag, SREG5), S (For signed tests Instruction Set, SREG4), V (Two's complement overflow indicator, SREG3), N (Negative Flag, SREG2), Z (Zero Flag, SREG1), and C (Carry Flag, SREG0). The slide footer includes the IIT KHARAGPUR logo and the NPTEL ONLINE CERTIFICATION COURSES logo.

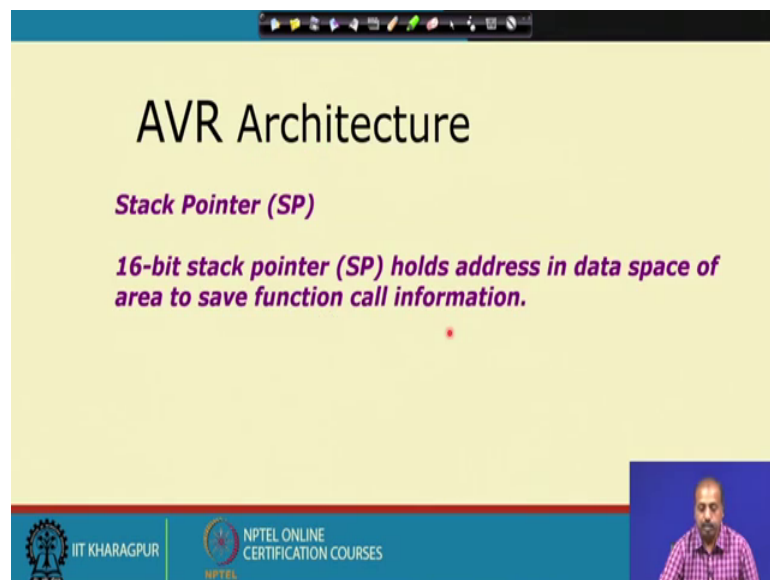
Then the status register. So, we have got a one status register SREG that has got 8 bit long flip flop and each bit 8-bit; 8-bit long register and each bit has got a different meaning. So, this I the most significant bit is the I bit which stands for global interrupt enable disable flag, ok. So, it is that I E or that E I beat or I E bit for different processors the global interrupts acma interrupt enable, ok, so, that bit.

Then the next bit is the T bit which is a transfer bit used by BLD and BST instructions. So, do will these are two special instructions that we will see later. Then H is the half carry flag. So, half carry flag is basically that auxiliary carry sort of thing. So, whenever there is a carry generated at the middle like if it is an 8-bit operation. So, from the fourth bit from third bit to four bit if there is a carry generated then it will be this half carry bit will be say.

Then there is S bit. So, this is for sign details to instruction set. So, this is this is a sign bit. So, if that raises if the content is sign negative then the S bit will be negative it will be 1 otherwise it will be 0. Then the V bit is a two's complement overflow indicator it is like when we are doing the operation so, there may be some overflow in the two's complement notation and representation.

So, that way it is that overflow information. Then we have got the negative flag. So, this is s so if the operation result is negative then this bit will be set. Then zero flag and carry flag. So, they have got the standard meaning this N, Z, C and V. So, they have got the normal meaning, but others they have got some special meanings.

(Refer Slide Time: 22:17)

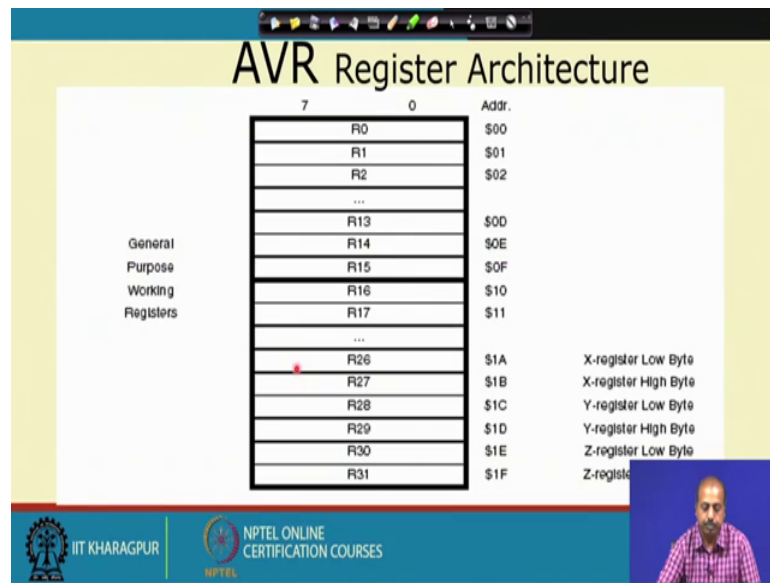


The image shows a presentation slide titled "AVR Architecture". The main heading is "AVR Architecture". Below it, the sub-heading is "Stack Pointer (SP)". The text on the slide states: "16-bit stack pointer (SP) holds address in data space of area to save function call information." There is a small red asterisk below this text. At the bottom of the slide, there are logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES. A small video inset of a speaker is visible in the bottom right corner of the slide.

So, then we have got the stack pointer. So, it has got the concept of stack. So, in R we did not have stack by default. So, we are here we have. So, it is a 16-bit stack point or it holds address or in the data space of area to save function call information. So, this is only holding the return addresses, ok. So, they will be saved in the stack and that is in the data space that will be saved in the data memory.

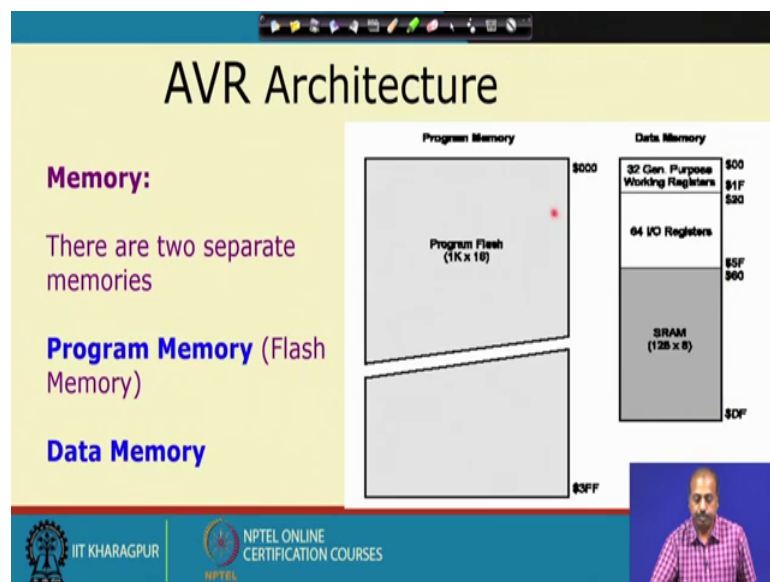


(Refer Slide Time: 22:44)



So, the register architecture is like this. So, so, we have got this address 00 to 1F. So, we have got 32 such registers R0 to R31 and out of that say this register number 26, 27, 28, 29, 30, 31 so, they are this x, y, z pointers.

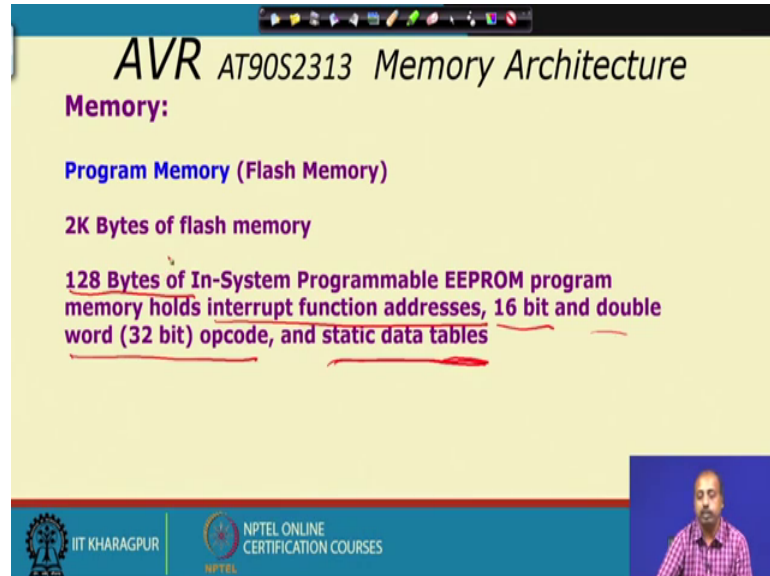
(Refer Slide Time: 23:11)



As far as memory is concerned so, memory can be thought to be consisting of two parts program memory and data memory. So, program memory is 1 kilo by 1 byte and this data memory is 64 it consists of that 32 general purpose working registers 64 I input

output registers and then we have got the SRAM part which is 128 locations of SRAM. So, that will be the data memory.

(Refer Slide Time: 23:45)



The slide displays the memory architecture for the AVR AT90S2313 microcontroller. It lists the following memory components:

- Program Memory (Flash Memory):** 2K Bytes of flash memory.
- EEPROM:** 128 Bytes of In-System Programmable EEPROM program memory holds interrupt function addresses, 16 bit and double word (32 bit) opcode, and static data tables.

The slide also features the IIT KHARAGPUR logo and NPTEL ONLINE CERTIFICATION COURSES branding at the bottom, along with a small video inset of the presenter.

So, program memory flash memory, so, it has got 2 kilobytes of flash there is outer then 128 bytes of in system programmable EEPROM program memory will be there. So, this is there, but this is used by the internal purpose. So, therefore, inter function address 16-bit and double word opcode. So, this so, this is 128 bytes so, they are used for this is 128 byte that we have so, they are used for interrupt function addresses that is that are the vector term is basically that inter vector table. So, that is there. So, it is 16-bit and 16-bit and double word opcode, or both 16-bit and 32 bit opcodes can be there and the static data table so, they can be there.

So, this is another indicator lattice whether we are going for 16-bit instruction or 32 bit instructions that will be also contained in that EEPROM and also we have to tear we have there may be some static data table. So, they can also be pay part of this in system program will be EEPROM, but actual program memory is the flash memory. So, that is the control the program that this microcontroller will run. So, that will be there in the flash memory. So, these are actually some for the 128 byte of EEPROM so, this is the these are for internal systems to usage and some static data tables can be put there.

(Refer Slide Time: 25:20)

**AVR AT90S2313 Memory Architecture**

**Data Memory**  
Used for data and is separate from the program memory.

**128 Bytes of SRAM**

- Registers reassigned the 32 Data Space addresses (\$00 - \$1F)
- I/O memory space contains 64 addresses for CPU peripheral functions such as, control registers, Timer/Counters, A/D converters and other I/O functions.
- I/O memory can be accessed directly or as the Data Space locations those of the Register File, \$20 - \$5F.
- Stack is effectively allocated in the general data SRAM, and consequently the stack size is only limited by the total SRAM size and the usage of the SRAM.

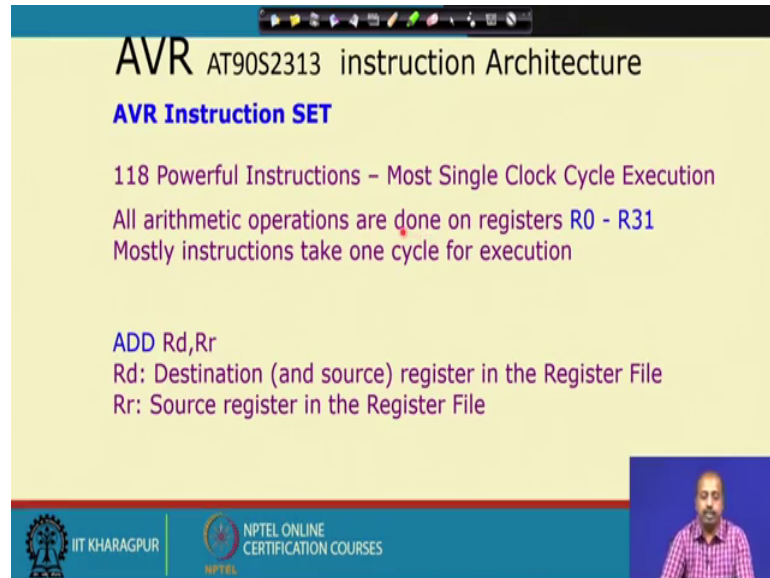
On the other hand that data memory so, it is used for data as naturally it is separate from program memory. So, we have got this Harvard architecture. So, program and data memories are separate then this 128 bytes of SRAM.

So, that is actually we are making this data memory. So, registers reassigned the 32 bit address space at 00 to 1F, so, that is 32 locations are though those registers will be holding this portion of the SRAM 32 locations are reserved for the registers then the I O memory space it will contain 64 addresses for the CPU peripheral functions. So, 64 locations are dedicated for these control registers, timer counter, AD converter other I O functions. So, 64 locations are reserved for that and this I O memory can be accessed directly or as data space locations and the locations those of the registers like dollar 20 to dollar 5F. So, you can either mention them as register or you can mention them as memory location so, this I O memory part.

So, this that is where you are accessing these timers AD converters and other I O routines are the other I O modules so that can be accessed both way and stack is effectively allocated in general data SRAM and the stack size is limited by the size of the SRAM. So, instance it is 128 byte out of that 32 bytes are gone for the registers 64 bytes are gone for the I O memory space. So, total 96 bytes are spent for this by the by the system. So, you are left with only 32 bytes for the step, ok.

So, this I so in this if it is if it is SRAM is limited to 128 bytes so, you can initialize this your stack pointer to location 97 and from that point onward yield stack will start and it can go up to 128 only.

(Refer Slide Time: 27:30)



The slide is titled "AVR AT90S2313 instruction Architecture" and "AVR Instruction SET". It lists "118 Powerful Instructions - Most Single Clock Cycle Execution" and states "All arithmetic operations are done on registers R0 - R31" and "Mostly instructions take one cycle for execution". It also shows the instruction format "ADD Rd,Rr" and defines "Rd: Destination (and source) register in the Register File" and "Rr: Source register in the Register File". The slide includes logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a speaker in the bottom right corner.

So, the AVR instruction set. So, it has got hundred 18 powerful instructions and most of the instructions they are single cycle execution instruction all arithmetic operations are done on registers R0 to R31 and mostly instructions take on once per cycle for execution. So, as we have already said that it will be all arithmetic operations are done on registered. So, R0 so, you cannot have R0 to R31 so, in some sense you can say they are memory locations as well because R0 to R31 is in the data ram. So, that way it is fine, but it is it is the operands are register operands, ok.

So, also this is similar to other processors that we have seen where this for example, in ARM processor we have said that all the operations are having registers are operand. So, here also the same philosophy is followed, but of course, you can say that, this is a RAM is also part of the processor. So, register and memory does not have any difference.