IN-CIRCUIT EMULATORS

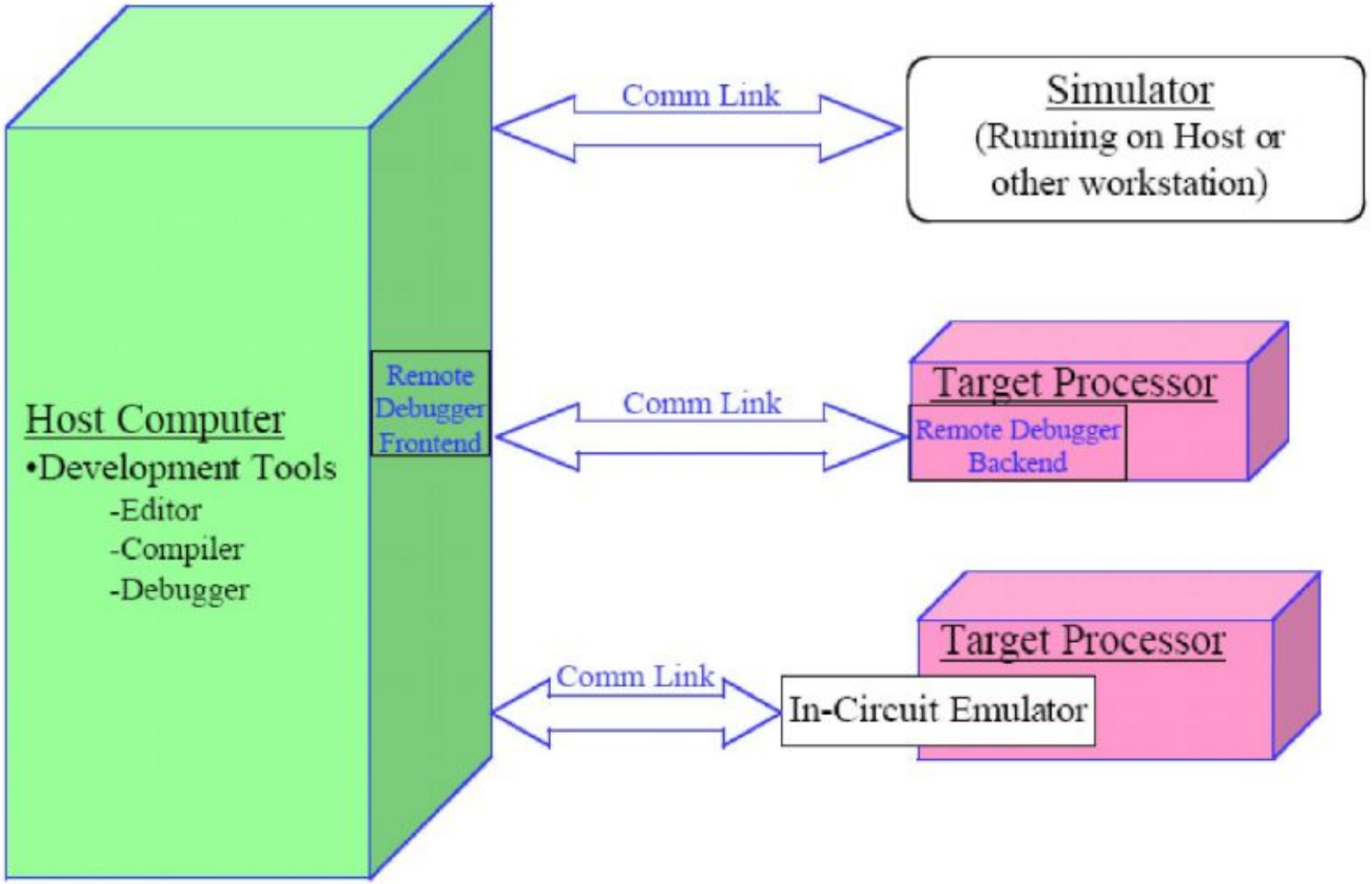
Dr.N.Arulanand,
Professor,
Dept of CSE,
PSG College of Technology

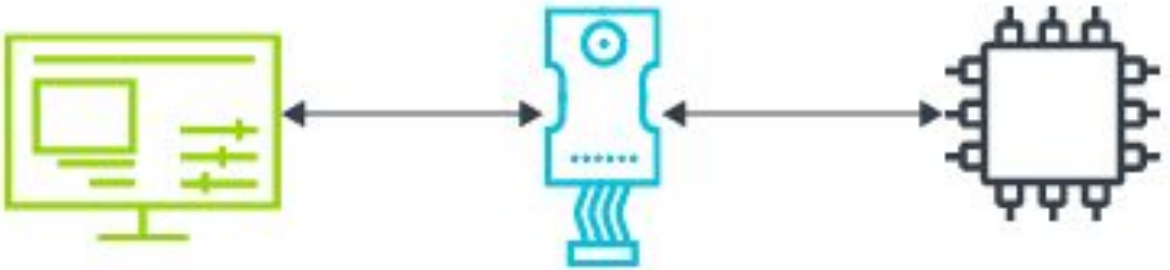
LET’S UNDERSTAND WHAT DEBUGGERS ARE FIRST

DEBUGGERS

- A debugger or debugging tool is computer program that is used to test and debug other programs
- A debugger or debugging tool is a computer program used to test and debug other programs (the "target" program).
- The main use of a debugger is to run the target program under controlled conditions that permit the programmer to track its operations in progress and monitor changes in computer resources that may indicate malfunctioning code.
- When a program crashes the debugger shows the actual position in the original code where the problem occurred, if it is a source level debugger.
- Low-level or Machine level debugger shows the line number where the problem is occurring.
- Most embedded systems do not have screen ,when a program fails it causes the processor to crash and lock-up.
- These errors have to be corrected and debuggers will help is rectifying the problem

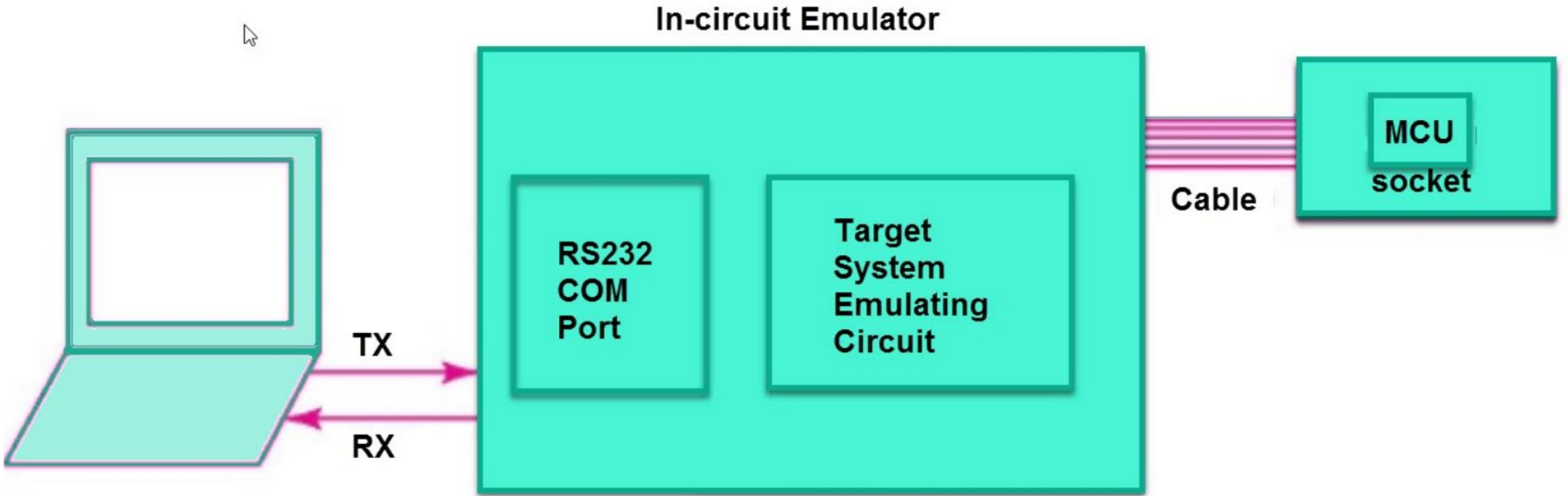
DEBUGGING TOOLS




Host machine with debugger
Debug Probe
Target being debugged

DEBUGGING WITH ICE

- Takes the place of the processor
- Contains copy of the target processor , plus RAM,ROM and its own embedded software
- Allows to examine the state of the processor while program is running
- Supports hardware and software breakpoints
- Stores information about each processor cycle that is executed



FEATURES AND LIMITATIONS OF ICE

FEATURES

- Run and test code in real time without target hardware.
- Observe and modify microprocessor registers and memory contents
- Execution breakpoints.
- Single stepping and source level debugging
- Trace program execution using internal logic analyzers
- What went wrong and where?

How In-Circuit Emulation Works?

1. A special **ICE debugger hardware** is connected between the **host computer** (where the code is written) and the **target embedded system**.
2. The **ICE tool replaces or intercepts** the microcontroller's execution, allowing real-time monitoring and debugging.
3. The **developer can set breakpoints, step through code, view registers and memory, and modify variables** while the system is running.
4. The **debugging data is sent to the host computer**, where a software debugger (e.g., **GDB, Keil, MPLAB, IAR Embedded Workbench**) provides analysis tools.

Key features of In-Circuit Emulation

Feature	Benefit
Real-Time Debugging	Observe how code executes on real hardware
Breakpoints & Stepping	Pause, step through code line by line
Register & Memory Access	View & modify RAM, ROM, I/O registers
Fault Analysis	Identify hardware/software bugs easily
Peripheral Emulation	Simulate missing hardware components

LIMITATIONS

- **AVAILABILITY AND COST**

Simple microprocessors are relatively easy to emulate, but there are several factors that reduce the potential markets and hence increase the cost

a. For even simple microcontrollers there are many variants so different ICE hardware is required for each

b. Modern cutting edge microprocessors are operating at very high speeds. To design emulators for these processors is very challenging and therefore extremely costly

c. Other tools are available that handle many microprocessor development problems at much less cost so many large design teams may only need one ICE (reducing sales). The danger of this situation is that when the real difficult problems arise that require an ICE to solve no one has taken the opportunity to learn how to use its capabilities.

LIMITATIONS

- **COMPLEX SETUPS**

Usual complaints from users are the emulators complexity. Too many developers never progress beyond the most basic of ICE features due to its complex setups.

LIMITATIONS

- **Transparency**

a. Physical : basically this implies that the ICE has to have the same pin-out and fits into the same socket as the microprocessor being emulated. This is usually achieved using a remote pod (the emulator) and a short (shorter the better) cable connector.

b. Electrical: implies that the drive and timing capabilities of the ICE are identical to the microprocessor being emulated. The extra circuitry with the ICE will degrade both the loading and timing. Any target system operating at the margin could "fall over" with an ICE. If the hardware is developed using the ICE then the system can almost be guaranteed to work when the ICE is replaced with a real microprocessor.

REAL TIME TRACE

REAL TIME TRACE

- Real Time Trace is one of the most important features of an emulator
- Trace captures the snapshots of executing code in a circular trace buffer – New processor states are override the old data.
- Trace is accompanied by the triggering mechanisms
- Trigger System determines when the trace memory should start capturing the trace
- Trace Systems runs continuously – enables the trace to record event prior to the trigger point

Event trigger system

- Event trigger system in an emulator is used to control the tracing of instruction flow and data flow and also stopping the execution of the processor
- Trigger event may be a single combination of processor states or a sequence of processor states
- The combination of ADDRESS, DATA, STATUS can be a single or sequence of events
- CHALLENGES
  - Pre-fetch queues
  - Cached processors

EXAMPLE OF REAL TIME TRACE FROM HP64700 EMULATOR

Trace List Offset=0 More data off screen (ctrl-F, ctrl-G)

Label: / Base:	Address hex	Data hex	Opcode or Status mnemonic	time count absolute
after	004FFA	2700	2700 supr data rd word	----------
+001	004FFC	0000	0000 supr data rd word	+ 520 nS
+002	004FFE	2000	2000 supr data rd word	+ 1.0 uS
+003	002000	2479	MOVEA.L 0001000,A2	+ 1.5 uS
+004	002002	0000	0000 supr prog	+ 2.0 uS
+005	002004	1000	1000 supr prog	+ 2.5 uS
+006	002006	2679	MOVEA.L 0001004,A3	+ 3.0 uS
+007	001000	0000	0000 supr data rd word	+ 3.5 uS
+008	001002	3000	3000 supr data rd word	+ 4.00 uS
+009	002008	0000	0000 supr prog	+ 4.52 uS
+010	00200A	1004	1004 supr prog	+ 5.00 uS
+011	00200C	14BC	MOVE.B #000,[A2]	+ 5.52 uS
+012	001004	0000	0000 supr data rd word	+ 6.00 uS
+013	001006	4000	4000 supr data rd word	+ 6.52 uS
+014	00200E	0000	0000 supr prog	+ 7.00 uS

OVERLAY MEMORY

Memory Overlay

Memory overlay is a technique used in **embedded systems and low-memory environments** to efficiently manage memory by **reusing the same physical memory space for different program segments at different times**.

Many microcontrollers and early computers have limited RAM/ROM.

Some functions or modules are only needed at certain times.

Instead of keeping all code in memory, only the necessary part is loaded and executed, saving space

How Memory Overlay Works?

- The system divides large programs into smaller code segments (overlays).
- At runtime, the required segment is loaded into a fixed memory location, replacing the previous segment.
- This allows multiple parts of a program to share the same memory space dynamically.

Imagine a **microcontroller with only 32 KB of memory** but a program requiring **64 KB**.

- The program is split into **Overlay 1** (Initialization), **Overlay 2** (Processing), and **Overlay 3** (Communication).
- **Only one overlay is loaded into RAM at a time**, reducing memory usage.

Memory Overlay in In-Circuit Emulation (ICE)

- Memory overlay plays a crucial role in **In-Circuit Emulators (ICE)** for debugging embedded systems.
- ICE allows developers to test software on real hardware while **overriding or modifying memory contents** dynamically.

How Memory Overlay Works in ICE

1. The **ICE hardware** intercepts memory accesses from the microcontroller (MCU).
2. Instead of reading from ROM, the ICE **maps an overlay memory (RAM or external storage) at the same address**.
3. The program runs as if the overlay memory is part of the actual system.
4. When debugging is done, the actual firmware can be burned into ROM/Flash.

Key uses of Memory Overlay in ICE

1 **Modify or Patch Firmware During Debugging**

- Developers can **replace specific ROM code sections** without physically changing the chip.
- Useful for **fixing bugs, testing new features, or inserting debugging hooks**.

2 **Simulate Different Program Scenarios**

- Different test cases can be loaded **without reprogramming the ROM/Flash memory**, speeding up testing cycles.

3 **Enable Breakpoints & Tracing**

- Since original ROM is **read-only**, ICE uses memory overlay to **redirect execution flow**, allowing **breakpoints, single-step execution, and code analysis**.

OVERLAY MEMORY

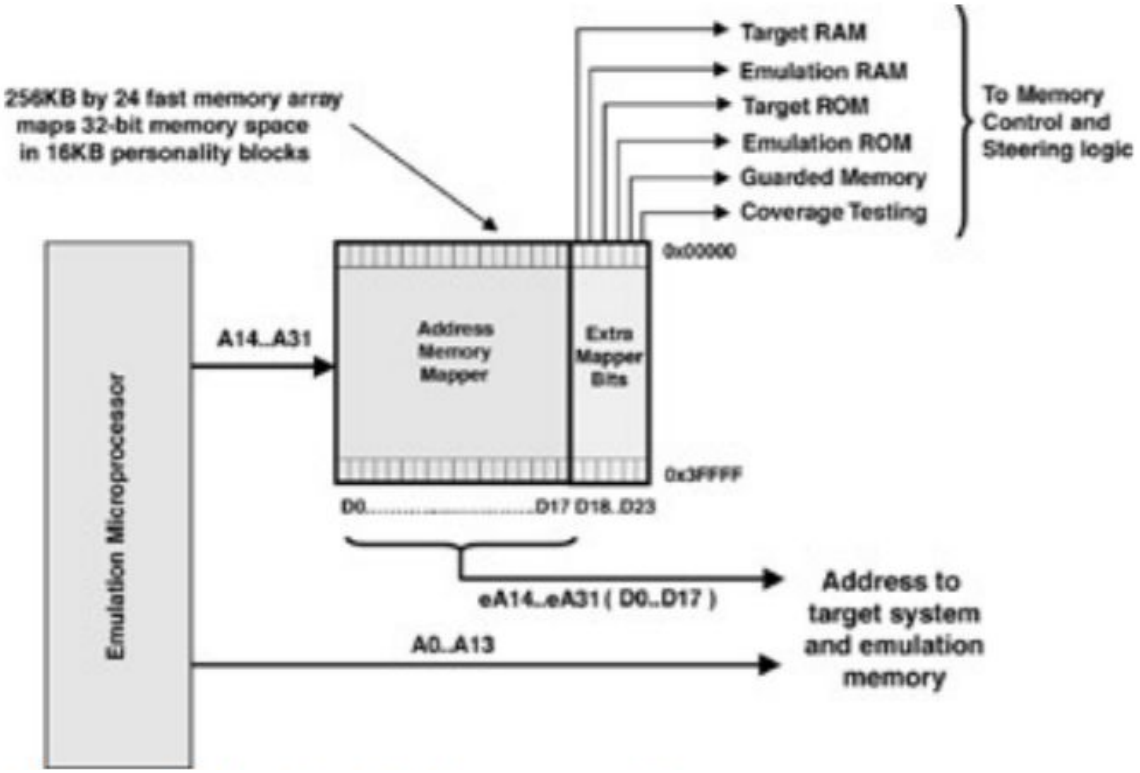
- Provides memory to overlay (replace) target systems memory resources or to temporarily fill memory regions with memory.

- Cover broad areas of the processor’s address space.

- Entire Memory Space can be mapped to provide memory regions in the memory anywhere in the address space of the target processor.

HOW OVERLAY MEMORY IS SETUP

- 18 address bit map the 18 higher order address bits, remaining six bits are use to assign personalities to each of the 16KB blocks.



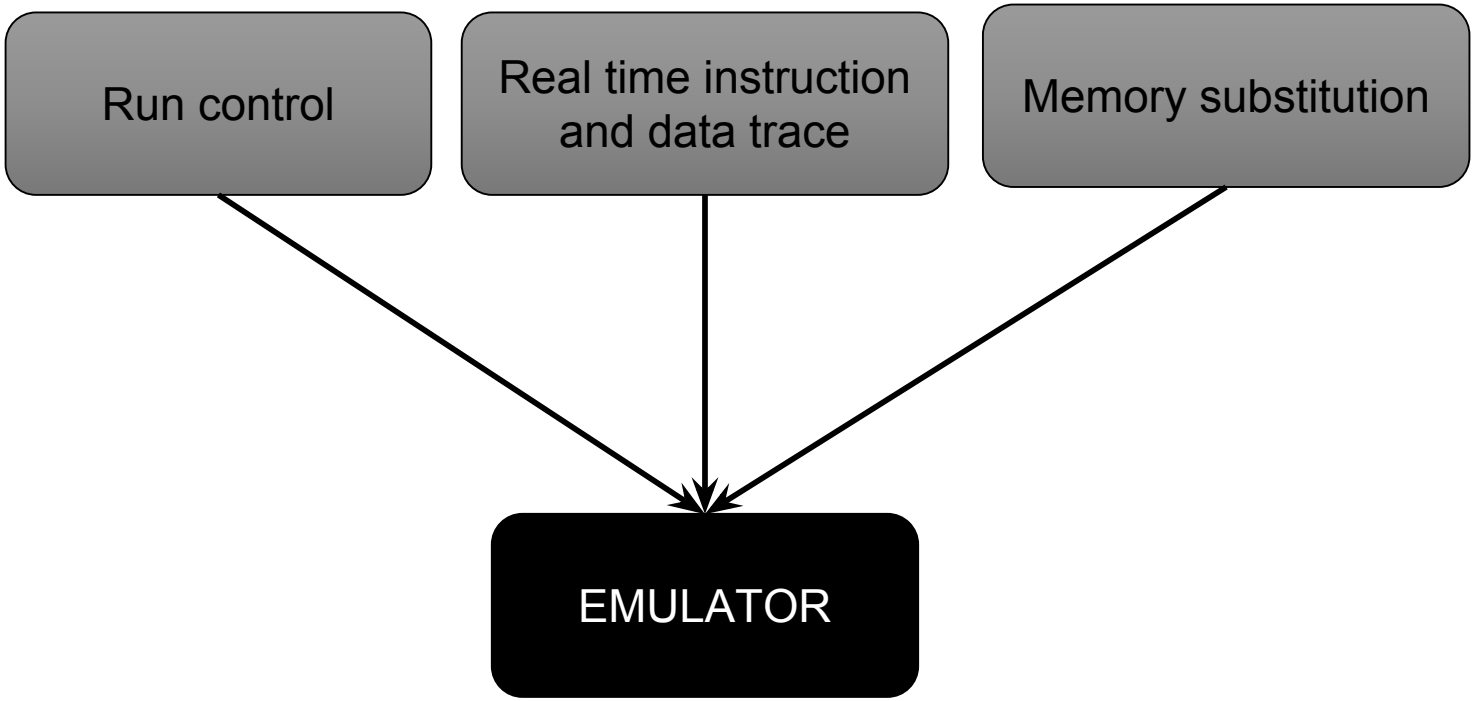
FEATURES OF OVERLAY MEMORY

- It can be given precedence over memory in the target system while substitution memory can only be used to replace a block of memory in the target system.

- Two other personality bits protects a region of memory from being written into even though it is assigned to the target or emulation RAM.

DISTRIBUTED EMULATOR AND ISSUES

DISTRIBUTED EMULATOR



***ICE can also be built with these three along with debug core instead debug kernel**

LIMITATIONS

- Emulator = Debugger + ROM Emulator + logic analyzer
- Drawbacks:
    - Cross triggering is difficult
    - Memory limitation
    - Break signal

USAGE ISSUES

- Emulator setup
- Knowledge of memory map
- Setting trigger

REFERENCES

- EMBEDDED SYSTEMS DESIGN: An Introduction to Processes, Tools, and Techniques- “ARNOLD S. BERGER”
- www.microcontrollertips.com
- https://educationalstuff1.tripod.com/ice.pdf
- YouTube