Embedded-Digital-Signal-Processing-Just-In-Time (EDSPJIT) Compiler

The Project
A small library for configurable sound processing, minus recompiling, reflashing and the overhead of conditionals

Runs on the embedded controller, compiles the given script immediately

The returned instructions are then executed from flash memory.

The Audience
Technical Musicians
Curious engineers to-be

The Tools
Tiva-C Series Evaluation Board
Keil IDE, C99, Vim & make

The Means
David Petrizze
USU ECE Department

Special Thanks
Family & Friends
Dr. Cripps & Jolynne Berrett

Conceptual Example
A basic mixer setup, with two signal inputs

Both inputs are mixed (added) before being sent to the first output. One input is forwarded directly to the second output.

Knobs control the volume of the signal (via multiplication) before being sent to each output.

Embedded C API

```c
void edspjit_Init(void);
void edspjit_Deinit(void);
void edspjit_AddInput(
  const char * name,
  const float * buffer);
void edspjit_AddOutput(
  const char * name,
  float * buffer);
void edspjit_Compile(
  const char * string,
  edspjit_Writer asm_out);
```

Alternatives include:

- **lots of pointers** (hardfaults, cache misses)
- **if statements** (pipeline flushes), and
- **hard to maintain code** for this level of configurability.

For further optimization, this concept may be extended to programming DSP chips in real time.

---

# Scripted in reverse polish
# notation!
Out1 := $In1 $Knob1 MUL;
Out2 := $In1 $In2 ADD $Knob2 MUL;

Just-In-Time Compiler

Audio In 1
Audio In 2
Knob 1
Knob 2

Code in Flash Memory

Audio Out 1
Audio Out 2

//\\\\\\ by David N. Petrizze III \\\\\\\ (dpetrizze@gmail.com) \\\\\\\