Error Handling Functions for the GBVM Library

File: `univ_errors.h`
Author: Soham Metha
Date: January 2025

The `univ_errors.h` file provides:

1. An `Error` enumeration for defining error codes used throughout the library.
2. Functions to display error messages, debug information, and handle program termination when errors occur.

Table of Contents

Error Enumeration

The `Error` enum defines various error codes to classify and identify specific types of errors.

Enum Value	Code	Description
`ERR_OK`	`0`	No error.
`ERR_STACK_OVERFLOW`	`1`	Stack overflow error.
`ERR_STACK_UNDERFLOW`	`2`	Stack underflow error.
`ERR_DIV_BY_ZERO`	`3`	Division by zero error.
`ERR_ILLEGAL_INST`	`4`	Illegal instruction error.
`ERR_ILLEGAL_INST_ACCESS`	`5`	Illegal instruction access error.

Enum Value	Code	Description
`ERR_ILLEGAL_OPERAND`	6	Illegal operand error.
`ERR_ILLEGAL_ALU_OPERATION`	7	Illegal ALU operation error.

Function Documentation

const char* errorAsCstr(const Error*)

Description:
Converts an `Error` enum value to a human-readable C-style string. If the `Error` value is unrecognized, the program crashes with the message:
`univ_errors : errorAsCstr : Unreachable`.

Parameters:

Parameter	Type	Description
`error`	`Error*`	Pointer to the `Error` value.

Returns:
A C-style string representation of the error.

void fileErrorDispWithExit(const char*, const char*)

Description:
Displays an error message alongside a file path and exits the program.

Parameters:

Parameter	Type	Description
`message`	`char*`	The error message to display.
`filePath`	`char*`	The file path associated with the error.

Returns:
None. Exits the program.

void executionErrorWithExit(const Error*)

Description:
Displays an execution error message based on the `Error` enum value and exits the program.

Parameters:

Parameter	Type	Description

Parameter	Type	Description
`error`	`Error*`	Pointer to the `Error` value.

Returns:
None. Exits the program.

---

void displayMsgWithExit(const char*)

Description:
Displays a generic error message and exits the program.

Parameters:

Parameter	Type	Description
`message`	`char*`	The error message to display.

Returns:
None. Exits the program.

---

void displayStringMessageError(const char*, String)

Description:
Displays a warning message alongside a `String` object. Formats the output to enhance readability.

Parameters:

Parameter	Type	Description
`msg`	`char*`	The warning message to display.
`str`	`String`	The associated `String` object.

Returns:
None.

---

void debugCommentDisplay(String*)

Description:
Displays debug comments from the given `String` object. These comments are identified by `#` or `;` and are formatted to a width limit of 125 characters.

Parameters:

Parameter	Type	Description
`s`	`String*`	The `String` object containing the debug comment.

Returns:
None.

Example Usage

Error Conversion Example

```c
#include "univ_errors.h"
#include <stdio.h>

int main() {
    Error err = ERR_DIV_BY_ZERO;
    printf("Error: %s\n", errorAsCstr(&err));
    return 0;
}
```

Handling File Errors

```c
#include "univ_errors.h"

void processFile(const char* filePath) {
    if (filePath == NULL) {
        fileErrorDispWithExit("File path cannot be null", filePath);
    }
}
```

Debug Comment Display

```c
#include "univ_errors.h"

void debugExample() {
    const char* s = "# This is a debug comment"
    String comment = (String){ strlen(s), s };
    debugCommentDisplay(&comment);
}
```

Notes

- Ensure `univ_strings.h` is included for the `String` type.
- Error handling functions are critical for debugging and graceful error recovery.
- Use `debugCommentDisplay` for debugging SASM files with properly formatted comments.