アセンブリ思考パズルゲーム (仮称: Core Logic) 企画構成案

個人開発・個人会社設立に向けた、デスクトップ向けロジックパズルゲームの設計図です。

1. コンセプト

- キャッチコピー: 「最小の命令で、最大の知能を構築せよ」
- 概要: プレイヤーは架空のCPUの低レイヤーエンジニアとなり、限られたレジスタ、スタック、メモリを操作して、与えられた入力から期待される出力（ゴール状態）を作り出す。
- ターゲット: プログラミング愛好家、パズルゲームファン、コンピュータの仕組みに興味がある層。

2. コア・メカニクス

基本ループ

1. 課題の確認: 「入力 A と B を足して出力せよ」「配列を昇順にソートせよ」などの目標が提示される。
2. コード記述: 独自の簡易アセンブリ言語を用いてプログラムを書く。
3. シミュレーション: ステップ実行（1行ずつ実行）または一括実行。
4. 検証: 最終的なレジスタやメモリの状態がゴールと一致すればステージクリア。

仮想アーキテクチャの要素

- **Registers (レジスタ):** ACC (アキュムレータ), R1, R2 などの高速な作業領域。
- **Stack (スタック):** PUSH, POP で操作する後入れ先出しのメモリ。
- **Memory (RAM):** アドレス指定でアクセス可能な広大なデータ領域。
- **Instruction Set (命令セット):**
    - MOV dest, src: データの移動
    - ADD, SUB, MUL: 算術演算
    - JMP, JZ, JNZ: ジャンプ・条件分岐
    - CMP: 比較

3. 画面構成 (UI/UX)

- コードエディタ (左側): シンタックスハイライト付きの入力エリア。行番号をクリックしてブレークポイントを設定可能。
- ステータスパネル (中央): - レジスタの値（16進数/10進数切り替え可能）。
    - 現在のプログラムカウンタ (PC)。
    - フラグレジスタ (Zero, Negative 等)。
- データパネル (右側):
    - メモリグリッド。
    - スタックの視覚的表示。

- デバッグコントロール (下部): 実行、停止、ステップ実行、リセット、実行速度スライダー。

4. ステージ構成 (プログレッシブ・ラーニング)

1. **Hello ASM:** MOV と OUT だけで値を転送する。
2. **Arithmetic:** ADD や SUB を使った計算。
3. **Control Flow:** JMP を使ったループの作成。
4. **Memory Access:** 大量のデータをメモリに格納し、ポインタで操作する。
5. **Subroutines:** スタックを使った関数呼び出しの再現。
6. **Advanced:** 最終的に、これまでの命令を組み合わせて「簡易的な計算機」や「暗号化プロセッサ」を構築する。

5. 技術スタック (Tauri + SvelteKit)

- **Frontend (SvelteKit + TypeScript):**
  - 高速なUI更新（レジスタの値の変化を即座に反映）。
  - CSSによる「レトロ・ターミナル」風の演出。
- **Backend (Rust):**
  - アセンブリのパース（解析）と実行エンジン。
  - 実行時間の計測やリソース制限（効率性の評価）。
- **Data:** ステージデータはJSON形式で管理。

6. 収益化・公開戦略

- 開発初期: itch.io で無料体験版（または早期アクセス）を公開し、フィードバックを得る。
- 本リリース: Steam にて 1,000円～1,500円程度で販売。
- 会社設立: リリース直前、または売上の目処が立った段階で法人化し、著作権と売上を管理する。

7. ビジュアルスタイル

- テーマ: ダークモード推奨。
- フォント: JetBrains Mono や Fira Code などのプログラミング用等幅フォント。
- 演出: ステップ実行時に、データがレジスタからメモリへ「流れる」ようなアニメーション。