Out-of-Order APEX Processor Design Document

1. Overview

This document details the design of a cycle-by-cycle simulator for an out-of-order processor based on the APEX ISA. The processor features 32 general-purpose registers (R0-R31), a condition code register (CC), and implements specialized control flow instructions JALP and RET. Key features include distributed reservation stations, unified register files, and a sophisticated control flow predictor.

2. Instruction Set Architecture

2.1 Control Flow Instructions

- **JALP (Jump and Link with Prediction)**
  - Format: `JALP <dest> #<signed literal>`
  - Operation: Saves return address in dest register and transfers control to PC + signed literal
  - Prediction: Always predicted taken
  - Final version of the program will store the return address in the return stack.
- **RET (Return)**
  - Format: `RET <src>`
  - Operation: Transfers control to address stored in src register
  - Uses 4-deep return address stack for prediction

2.2 Other Instructions

- Conditional Branches: BZ, BNZ, BP, BN
- Memory Operations: LOAD, STORE, LTR, STR
- Arithmetic/Logic: Standard integer operations
- Multiplication: MUL instruction

3. Processor Architecture

3.1 Functional Units

1. **Integer Functional Unit (IntFU)**
   - Single stage execution
   - Handles: Integer operations, branches, JALP, JUMP, NOP
   - Verifies branch predictions and RET addresses
2. **Multiplication Unit (MUL)**
   - 4-stage pipeline
   - Dedicated to MUL instructions
   - Result available after last stage

3. **Memory Unit (MEM)**
    - 3-stage pipeline
    - Stage 1: Address calculation
    - Stages 2-3: Memory access
    - No store-to-load forwarding within LSQ

3.2 Pipeline Stages

1. **Fetch (F)**
    - One cycle operation
    - Control flow predictor lookup
    - Instruction fetch from memory
2. **Decode (D1, D2)**
    - Two-cycle operation
    - D1: Initial decode, predictor entry creation
    - D2: Renaming, dispatch preparation
3. **Issue/Execute**
    - Distributed to appropriate reservation station
    - Function unit execution
    - Forwarding support
4. **Commit (C)**
    - In-order commitment via ROB
    - One cycle delay for commitment
    - Register file updates

3.3 Reservation Stations

1. **Integer Reservation Station (IRS-8)**
    - 8 entries
    - Handles IntFU operations
    - Timestamp-based issue prioritization
2. **Load/Store Queue (LSQ-6)**
    - 6 entries
    - Serves as MEM reservation station
    - Address calculation fields
3. **Multiplication Reservation Station (MRS-2)**
    - 2 entries
    - Dedicated to MUL operations
    - Timestamp-based issue prioritization

3.4 Register Management

1. **Unified Physical Register File (UPRF)**
    - 60 entries for general-purpose registers
    - FIFO-based free list allocation
    - Renamer deallocates policy
2. **Unified Condition Register File (UCRF)**
    - 10 entries for CC flags
    - Stores Z, N, P flags
    - FIFO-based free list allocation
3. **Rename Tables**
    - Front-end rename table for dispatch
    - Back-end rename table for commitment
    - Checkpointed for control flow recovery

4. Control Flow Predictor

4.1 Predictor Structure

- 8-entry FIFO predictor table
- Fields per entry:
    - Instruction type (Branch/JALP/RET)
    - Target address (for branches/JALP)
    - Return stack pointer (for RET)
    - Prediction information (for branches)

4.2 Return Address Stack

- 4-deep stack implementation
- Push: During JALP execution
- Pop: During RET prediction
- Checkpointed for misprediction recovery

4.3 Prediction Policies

- Negative offset branches: Always taken
- Positive offset branches: Use last execution outcome
- JALP: Always taken
- RET: Use return address stack

4.4 Misprediction Handling

1. **Checkpoint Structures**
    - Rename tables (front-end and back-end)
    - UPRF/UCRF free lists and valid bits

    - Return address stack
    - BIS (Branch Information Storage)

2. **Recovery Actions**

    - Pipeline stall
    - Flush mispredicted path instructions
    - Restore checkpointed state
    - Update predictor information

5. Timing Specifications

5.1 Pipeline Timing

- F, D1, D2, Dispatch : One cycle each
- IntFU: Single cycle
- MUL: Four cycles
- MEM: Three cycles
- Commit: One cycle with one cycle delay

5.2 Forwarding

- Tag broadcast: One cycle
- Value broadcast: One cycle
- Early tag broadcast for simplified timing
- Register read at issue time

6. Key Structures and Functions

6.1 Parser

Structure Fields:

- Input file name (string)
- List of parsed instructions
- Symbol table for labels

Associated Functions:

- **ParseFile**: Reads assembly file and converts to internal instruction format
- **ValidateInstruction**: Checks instruction syntax and format
- **GetNextInstruction**: Returns next parsed instruction

6.2 CPU

Structure Fields:

Registers:

- UCRF (Unified Condition Register File)

- UPRF (Unified Physical Register File)
- Front-end rename table
- Back-end rename table

Queues and Buffers:

- ROB (80 entries)
- IRS (8 entries)
- MRS (2 entries)
- LSQ (6 entries)

Control Flow Structures:

- Predictor table (8 entries)
- Return address stack (4 entries)
- Branch Information Storage (BIS)

Pipeline Stages:

- Fetch (F)
- Decode stages (D1, D2)
- Dispatch stages (IRS, LSQ, MRS)
- Function units (IntFU, MulFU, MemFU)
- Commit stage (C)

Associated Functions:

- **Initialize**: Sets up initial CPU state
- **SimulateCycle**: Advances simulation by one cycle
- **CheckpointState**: Creates checkpoint of CPU state
- **RestoreState**: Restores from checkpoint
- **UpdateArchitecturalState**: Commits instruction results
- **HandleMisprediction**: Manages control flow recovery

6.3 Predictor

Structure Fields:

- FIFO prediction queue (8 entries)
- Return address stack (4 entries)
- Prediction history
- Branch target buffer

Associated Functions:

- **PredictControlFlow**: Makes prediction for control flow instructions
- **UpdatePredictor**: Updates prediction based on actual outcome
- **PushReturnAddress**: Manages return address stack pushes

- **PopReturnAddress**: Manages return address stack pops
- **CheckPrediction**: Verifies prediction accuracy
- **HandleEmptyStack**: Manages empty return stack condition

6.4 Reservation Stations

IRS (Integer Reservation Station)

Fields:

- Instruction queue entries (8)
- Ready flags
- Operand values/tags
- Timestamp fields

Functions:

- **AllocateEntry**: Assigns new instruction to station
- **UpdateOperands**: Handles operand availability
- **SelectReadyInstruction**: Chooses instruction for issue
- **ManageTimestamps**: Tracks instruction age

LSQ (Load Store Queue)

Fields:

- Address calculation fields
- Memory operation type
- Dependency tracking
- Store data buffer

Functions:

- **CalculateAddress**: Computes memory addresses
- **CheckDependencies**: Verifies memory operation safety
- **ProcessMemoryOperation**: Handles load/store execution
- **ManageQueueOrder**: Maintains memory ordering

MRS (Multiply Reservation Station)

Fields:

- Instruction queue entries (2)
- Operand status tracking
- Timestamp fields

Functions:

- **AllocateMultiplyOp**: Assigns multiply operations
- **TrackOperandStatus**: Monitors operand availability
- **SelectForExecution**: Chooses ready multiplication
- **ManageTimestamps**: Tracks instruction age

6.5 Register Management

UPRF (Unified Physical Register File)

Fields:

- Physical registers (60 entries)
- Free list
- Valid bits
- Mapping table

Functions:

- **AllocateRegister**: Assigns physical register
- **DeallocateRegister**: Returns register to free list
- **UpdateMapping**: Manages register renaming
- **ReadRegister**: Retrieves register value
- **WriteRegister**: Updates register value

UCRF (Unified Condition Register File)

Fields:

- Condition registers (10 entries)
- Flag bits (Z, N, P)
- Free list
- Valid bits

Functions:

- **AllocateFlags**: Assigns physical flag register
- **DeallocateFlags**: Returns flags to free list
- **UpdateFlags**: Sets flag values
- **ReadFlags**: Retrieves flag values

6.6 Reorder Buffer (ROB)

Structure Fields:

- Circular buffer (80 entries)
- Completion status
- Result values
- Destination registers

- Control flow information

Associated Functions:

- **AddInstruction**: Enters new instruction
- **UpdateStatus**: Tracks instruction completion
- **CommitInstruction**: Updates architectural state
- **FlushBuffer**: Handles misprediction recovery
- **TrackDependencies**: Manages instruction dependencies

6.7 Branch Information Storage (BIS)

Structure Fields:

- Active branch entries
- Prediction information
- Checkpoint references
- Target addresses

Associated Functions:

- **TrackBranch**: Records branch information
- **UpdateBranchStatus**: Maintains branch state
- **HandleMisprediction**: Manages recovery process
- **CleanupEntries**: Removes completed branches

6.8 Pipeline Control

Structure Fields:

- Stage status registers
- Stall conditions
- Flush signals
- Forward paths

Associated Functions:

- **ManagePipeline**: Controls pipeline flow
- **HandleStalls**: Manages pipeline stalls
- **ProcessForwarding**: Controls data forwarding
- **FlushPipeline**: Handles pipeline flushes
- **SynchronizeStages**: Coordinates pipeline stages

7. Design Considerations

7.1 Simplification Choices

1. Single branch/RET in execution
2. Early tag broadcast

3. FIFO-based allocation
4. No LSQ forwarding

7.2 Error Handling

1. Misprediction recovery

10. Implementation Schedule

10.1 Phase 1: Basic Infrastructure

- Pipeline stages
- Register management
- Basic instruction execution

10.2 Phase 2: Out-of-Order Features

- Reservation stations
- Reorder buffer
- Register renaming

10.3 Phase 3: Control Flow

- Predictor implementation
- Return stack
- Misprediction recovery