AMULET3 . . . Revealed

amulet (n.) a chARM; a medicine supposed to have occult operation.

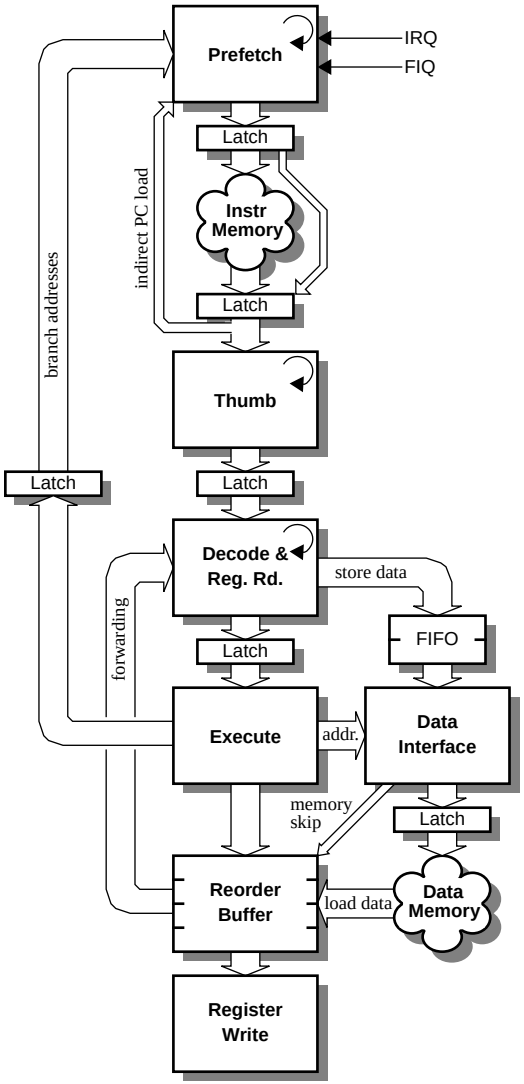
Chambers 20th Century Dictionary

Fillet of a fenny snake,
In the cauldron boil and bake;
Eye of newt and toe of frog,
Wool of bat and tongue of dog,
Adder's fork and blind-worm's sting,
Lizard's leg and owlet's wing,
For a charm of powerful trouble,
Like a hell-broth boil and bubble.

Macbeth Act 4 Scene 1

What is AMULET3?

- An ARM implementation
  - ARM “level 4” (ARM9) instruction set
  - “Thumb” mode
- Fully asynchronous
  - Low power
  - Low EMI
  - Automatic halt mode
- High performance (for an ARM)
  - equal to an ARM9 (?), using the same
  - generic 0.35µm, 3 layer metal process



Architectural features

- Branch prediction
- Unwanted cycle suppression
- Thumb decoder
- Unrestricted register forwarding
- Load/store with out-of-order completion
- Dual ("Harvard") bus interface
- Support for precise exceptions

Prefetch unit

Asynchronous features

- Non-deterministic prefetch depth
- *NEW* Branch/Interrupt fetch suppression (skips memory cycle)
- Variable cycle time (BTB optimised for sequential cycles)
- Halt

Low power features

- *NEW* Branch/Interrupt fetch suppression (skips memory cycle)
- Split BTB (as AMULET2)
- *NEW* “2 for 1” fetches in Thumb mode

Novel features

- *NEW* Interrupts processed here (low latency)
- *NEW* Indirect branches executed here

Thumb Decoder

ARM mode (interrupt/predicted branch/...)

- Pass instruction

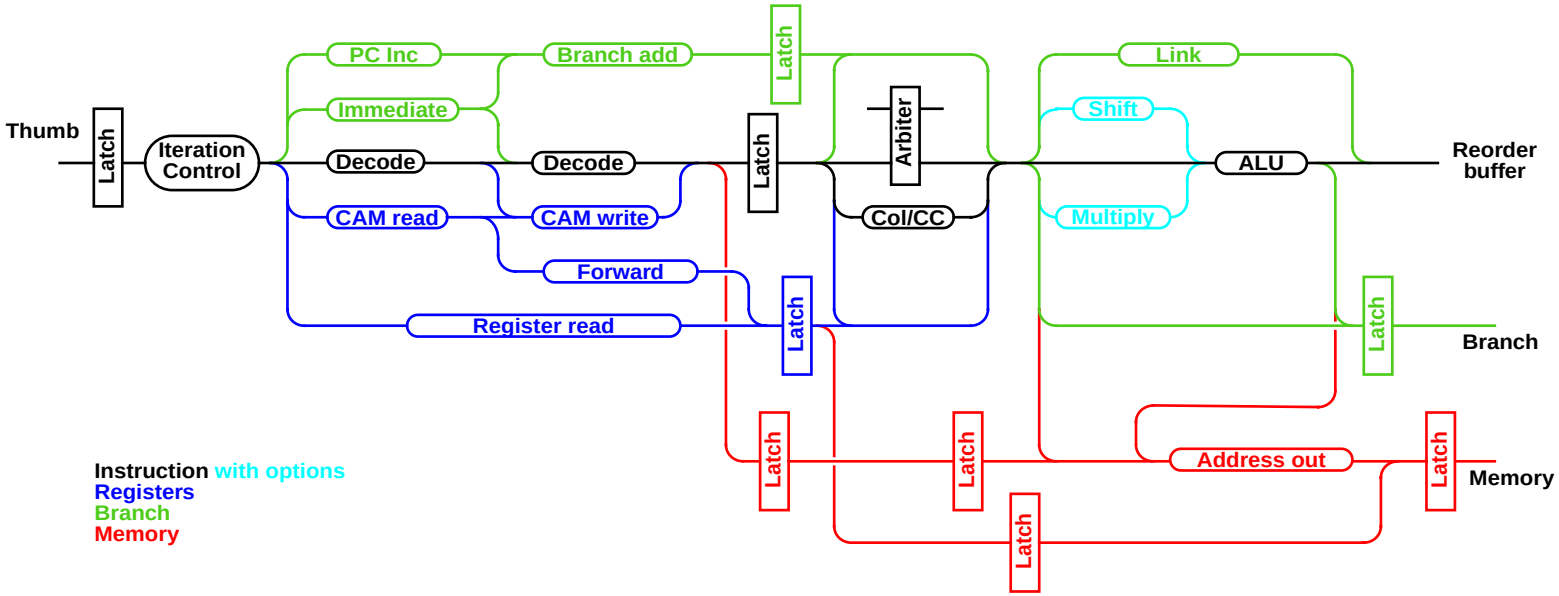
Thumb mode

- Expand 32-bit word into two full instructions
- Each instruction takes time to generate
- Usually two 16-bit instructions are expanded into two full ARM instructions

Asynchronous features

- Variable delay
- “One in, two out” handshakes

Decoding & Execution



Features

- Considerable parallelism
- Skewed pipeline latches
- Extra buffer for LDM/STM

Instruction Decoder/Register Read

Asynchronous features

- Multi-cycle operations for load/store multiple and “long” multiplication
- Branch calculation done here (for later speed-up)
- Fast discard of instructions in branch “shadow”

Low power features

- Only required buses/decoders activated

Novel features

- First use of asynchronous reorder buffer (we believe!)

Instruction Decode – example

LDMNEIA SP!, {R0..R10,PC} ; (Conditional) pop registers and return

- Requires 1 register read (SP) and 13 register writes
- Decoded into 12 cycles (one if condition fails)
- First decode cycle stretched:
  - to identify first register in list
  - to allocate two destinations
  - to handshake condition test result back from exec. unit
- First cycle loads R0 and PC in “parallel” – lowers latency
  - (also writes back modified SP)
- Other cycles only sent to memory interface

(Last cycle merely to ensure abort function correct)

Execution unit

Asynchronous features

- Fast discard of instructions in branch “shadow” or condition failure
- Operation dependent execution time (optimised for commonest operations)
- Extra time inserted when required for shift (ARM requires shifter in series with ALU)
- Self-timed, iterative multiplier – invoked when required
- Short cycle for branches (precalculated)
- Short cycle for CMP et al. (no result written)
- Arbiter allows interruption by memory fault

Memory interface

Asynchronous features

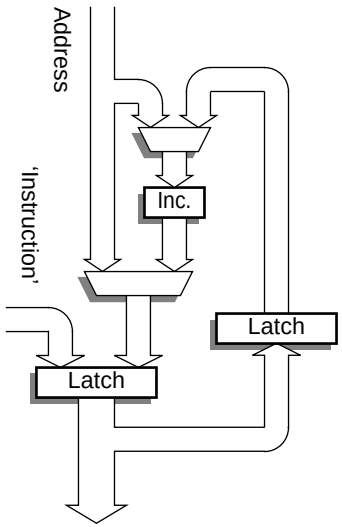
- Fast bypass for “failing” instructions
- Assumes unified, dual-port memory – any arbitration/ sequencing handled externally

Low power features

- Semi-autonomous address generation
- Sequential addresses (good for DRAM optimisation)

Novel features

- Out-of-order completion
- Support for coprocessor interface
- Support for ARM debug interface



Performance

These figures are based on a “typical” process and extracted layout using EPIC Timemill

Instruction timings (& throughput):

- MOV 7.5ns (130MHz)
- ADD 8.7ns (115MHz)
- TST 7.2ns (140MHz)

No dependency penalty (so far) observed after data operations

- LDM ! 9.8/7.7ns (100/130MHz) first/subsequent cycles

(Note: these numbers do not include any limitations due to memory)

This compares favourably with ARM9 (120MHz on same process)

(Almost) all instructions take one cycle –
although some cycles are longer than others

Improvements

The performance improvements over AMULET2e are approximately:

Feature	Improvement
Architecture (CPI)	+10%
Architecture (memory*)	not yet quantified
Cycle time	+80%
Process	+40%
Total (approx.):	+180% (+)

*The memory system is effectively dual-ported

We expect AMULET3 on a 0.35µm process to be almost 3x faster than the 0.5µm AMULET2e

About a factor of two improvement is from the design alone

Conclusions

AMULET has now caught up with the synchronous ARM

- Smaller – Certainly
- Faster – Maybe
- Lower power – Possibly
- Lower EMI – Probably

- Y2K compliant – no calendar – no clock

- Availability – Real Soon Now!

www.cs.man.ac.uk/amulet/AMULET3_uP.html