Admin stuff

- Course selection for period 2
  - Deadline: Oct 9
- Exam signup (<u>mandatory)</u>
  - Deadline: Oct 11
- A first this year: <u>digital exam</u>
  - Exam carried out in exam hall, but at keyboard
  - Will provide a “training exam”

Software vs Hardware in embedded electronics

DAT093

lars.svensson@chalmers.se

Outline

- Software from hardware point-of-view
- Software vs hardware: compare and contrast
- Case study: from hardware-centric to software-centric development

Electronics vs. software

- Every non-trivial electronic system today has a software component
- Software often dominates / determines…
  - …performance
  - …development time
  - …total development cost
- To some people, electronics <u>is</u> software :-/

Why use software?

- Hardware reuse <u>within</u> project
    - One adder, many addition operations
- Processor is extremely flexible component
    - Programmability enables hardware reuse <u>across</u> projects
- Highly capable development environments
    - Compilers, debuggers, simulators, …

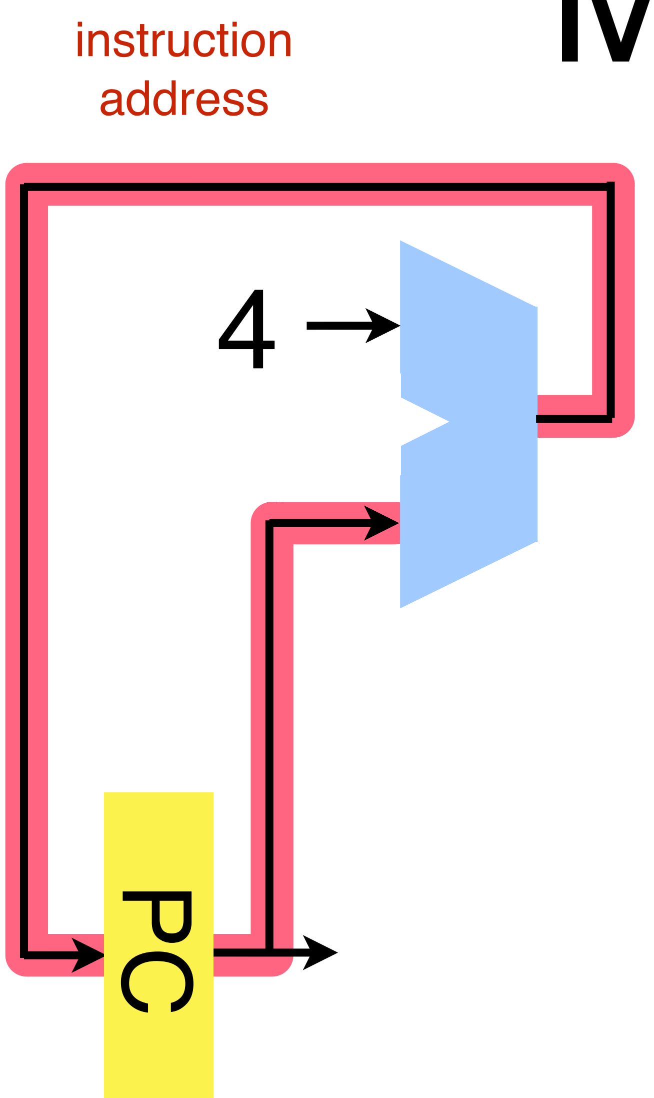
Low Non-Recurrent Engineering (NRE) cost!

What's a processor?

- A component that operates on data according to a stream of instructions

- Arithmetic and logic unit(s)
- Memory / registers
- Control
  - Program counter
  - Instruction fetch / decode
  - Data buses / muxes

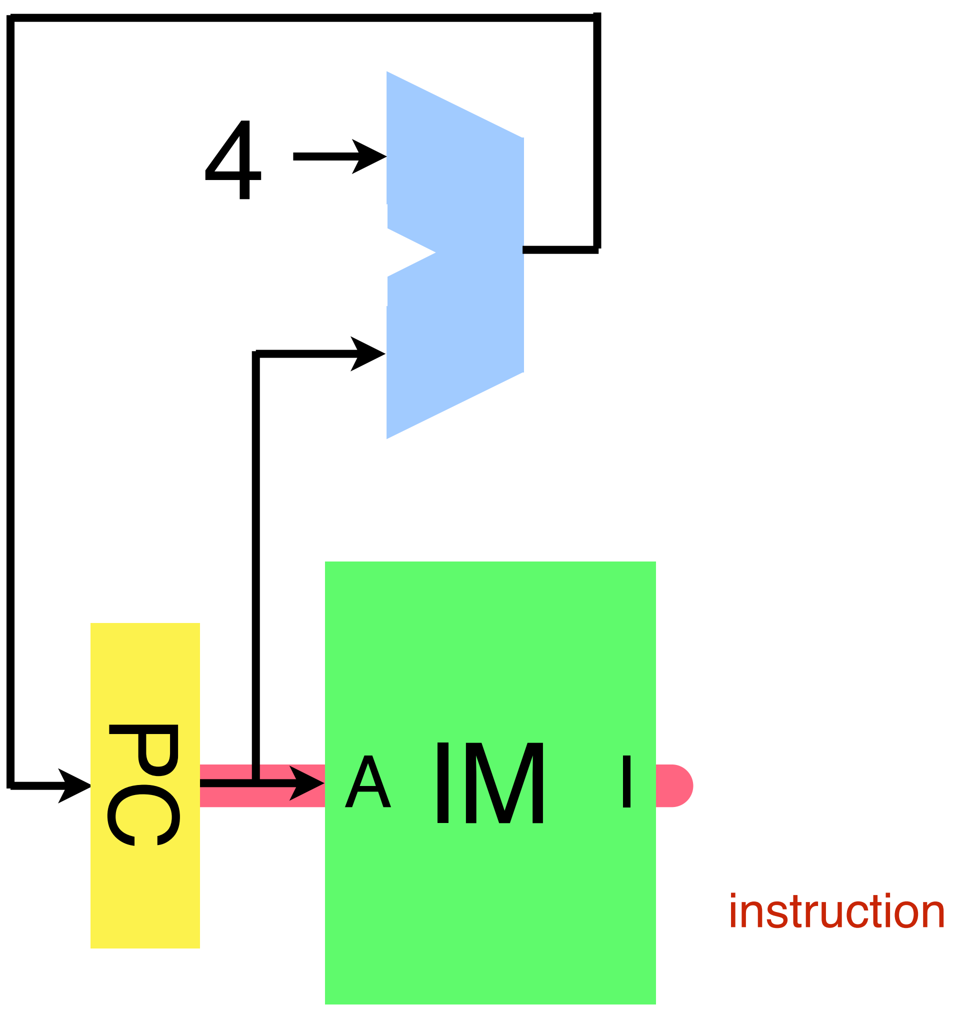
MIPS example: PC



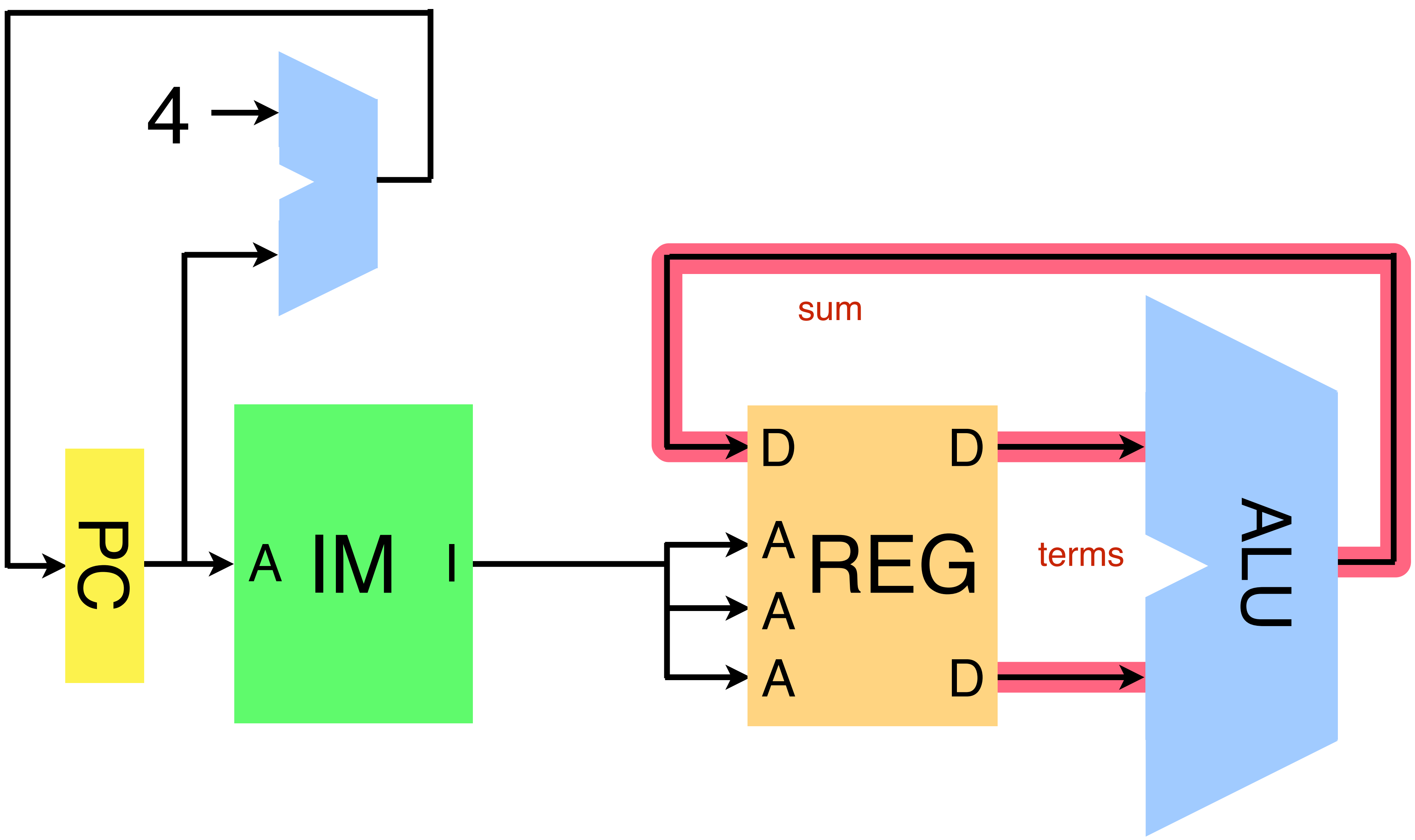
EDA332

[Patterson, Hennessy. Computer organization and design. 4th Ed. 2011]

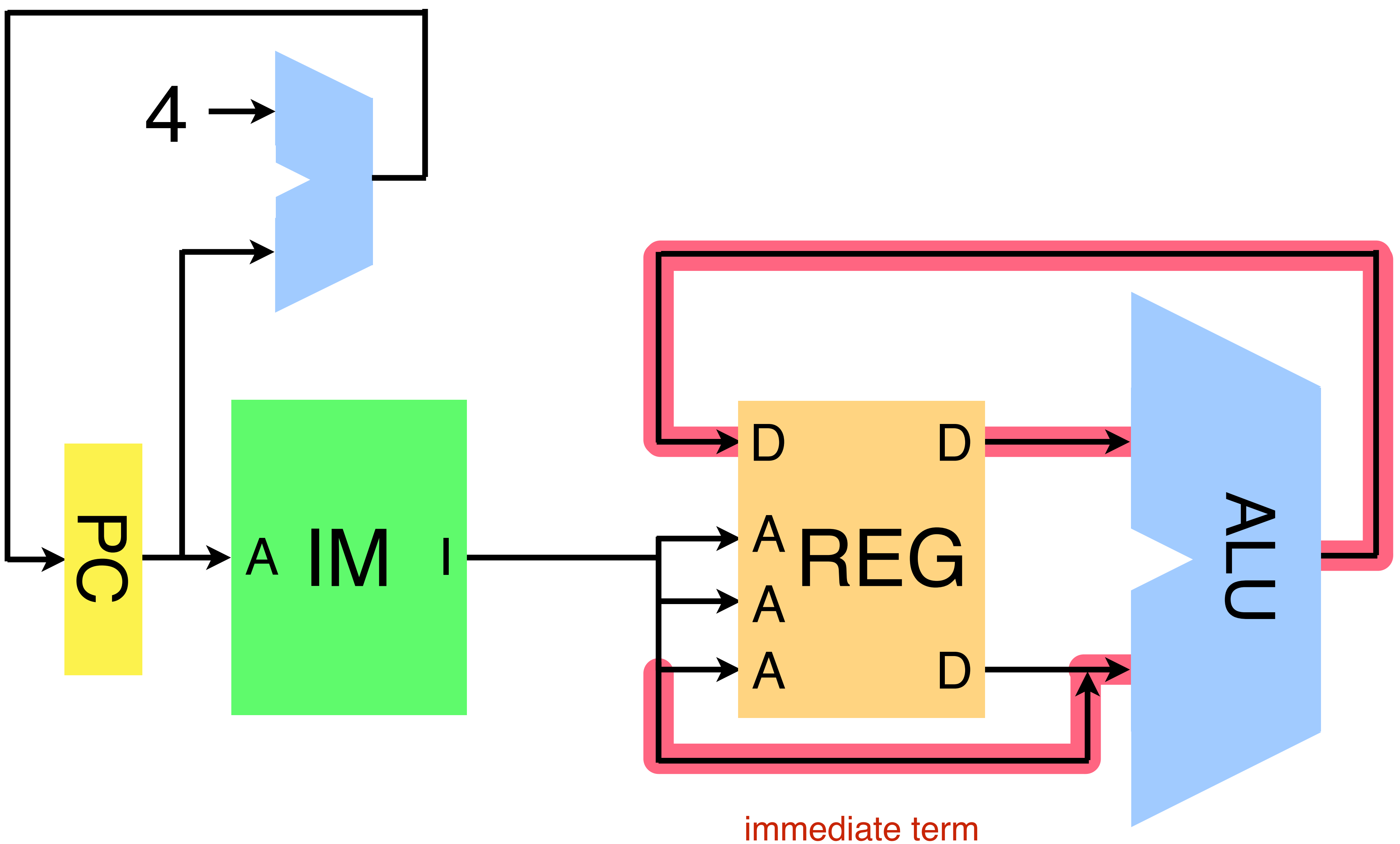
IF



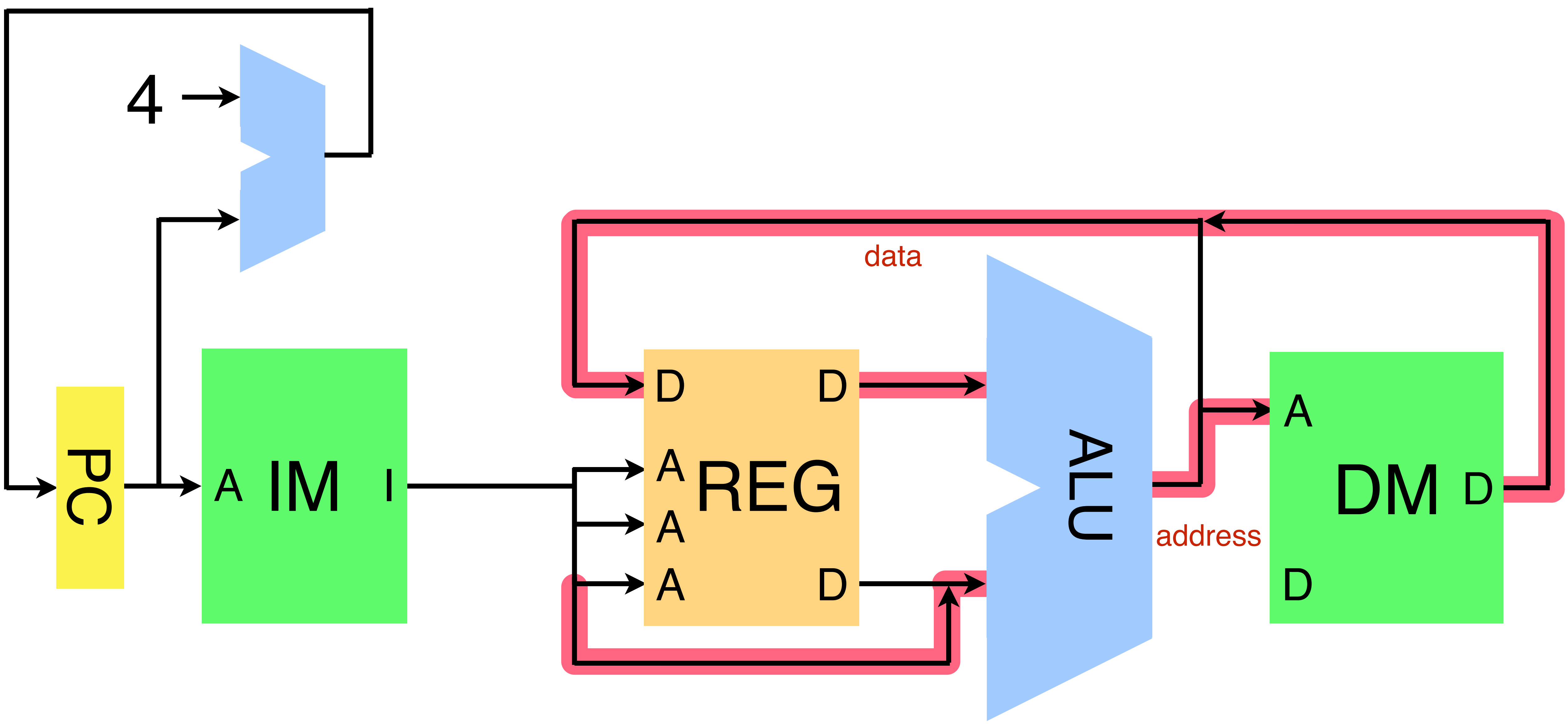
add $1, $2, $2



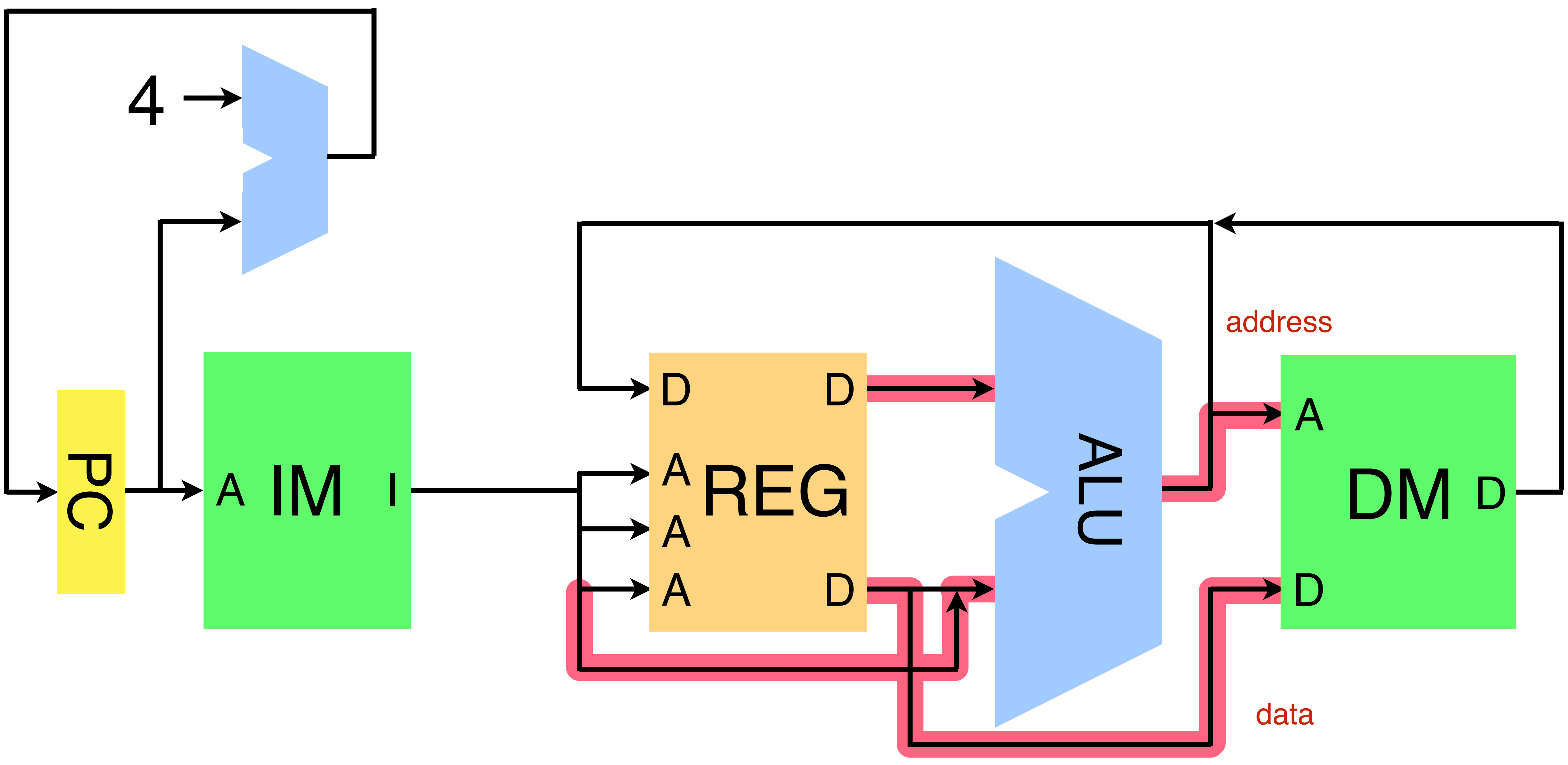
addi $1, $2, 55



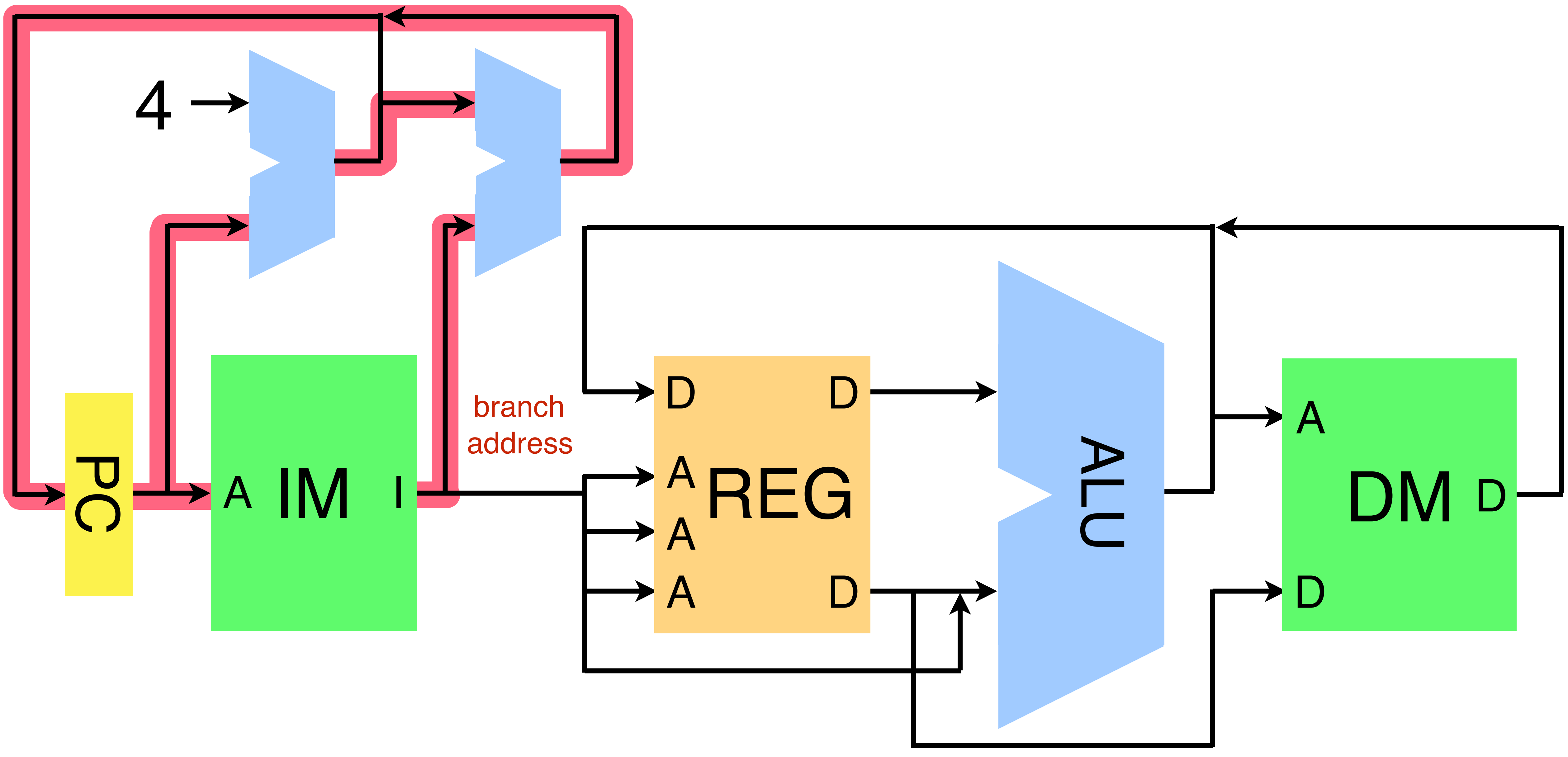
lw $t0, 8($s3)



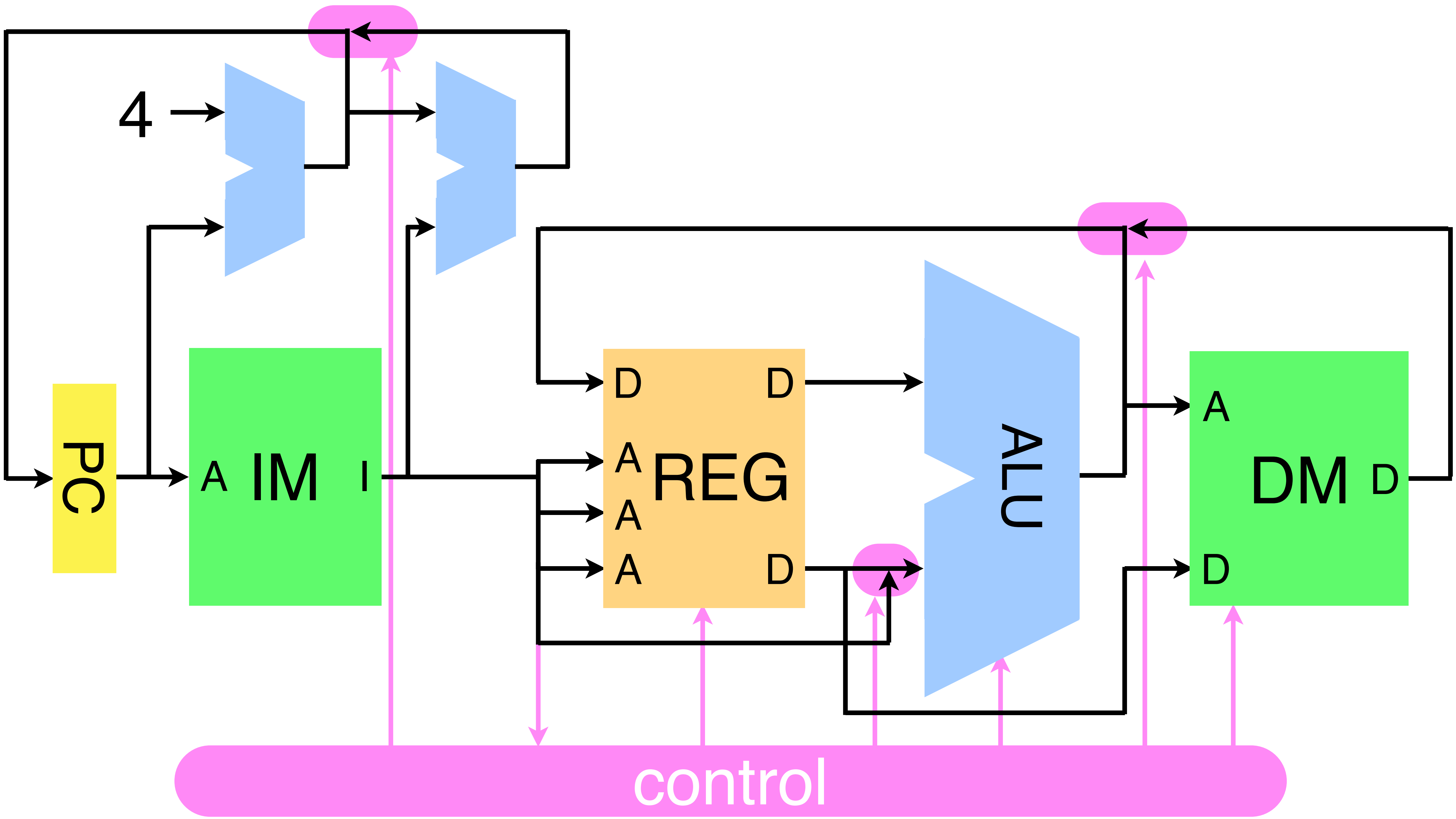
sw $t1, 8($s3)



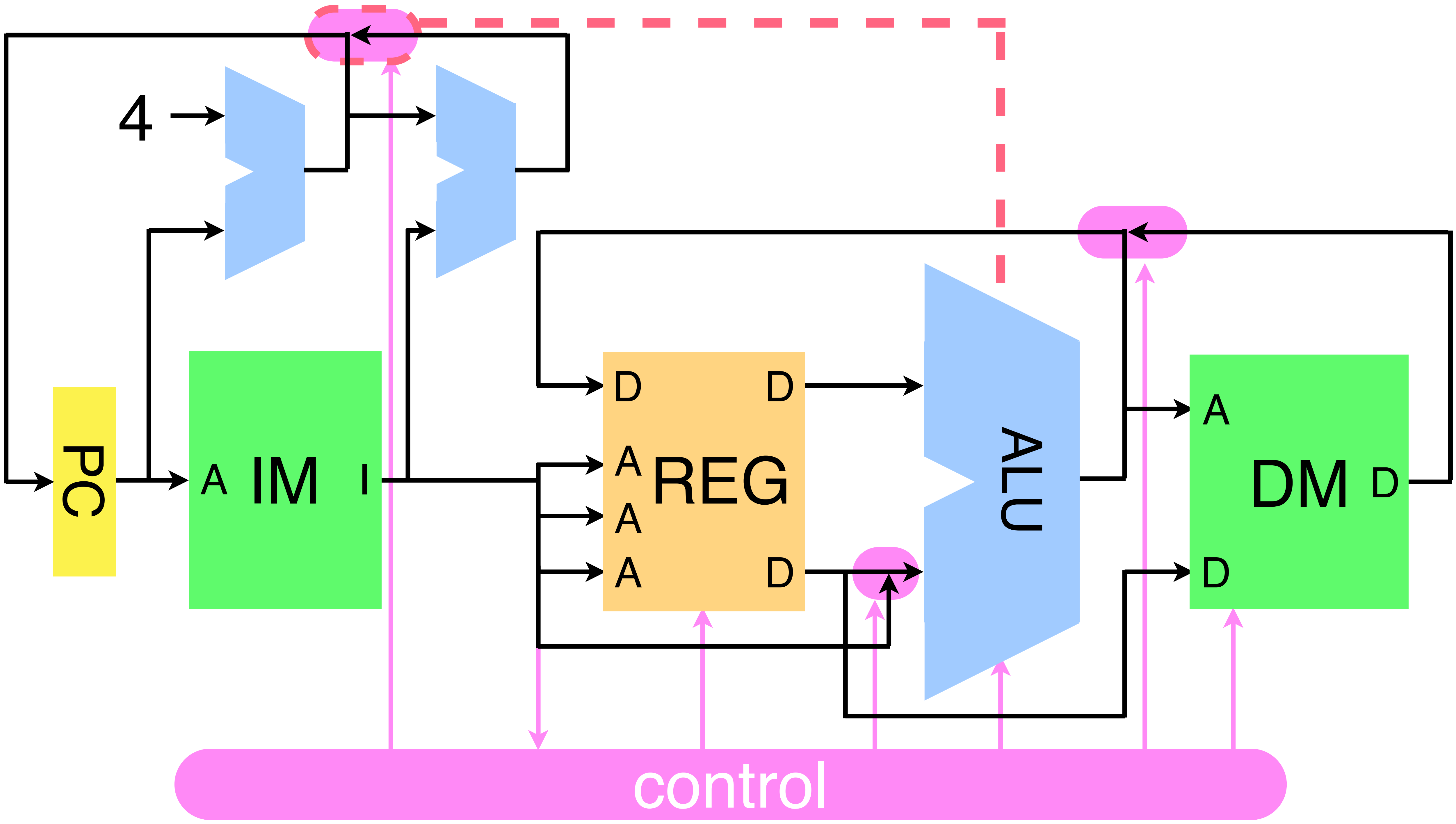
branch



control logic



ctl



Simple processor

- Single ALU
- Single register file
- Two one-level memories (instructions/data)
- Single instruction / cycle
- All these properties can be improved on!
- Pipeline to keep several instructions in flight (improved performance)

Performance?

- Def: Operations per unit time
- Improve by:
  - High operational frequency
  - Several ALUs
- Operations need operands
  - High performance needs buses, registers, memory ports
- Processor needs instructions and data
  - Caches, main memory

DAT105

EDA284

Informal processor classes

"Impure" examples, combinations, etc.

- Microprocessor
  - Number of cores?
- Microcontroller (processor + memory + peripherals)
  - Common in embedded systems
- Signal processor (focus on vector, matrix ops; scalar product, multiply-acc, a = a + b * c)
  - Number of cores?
- Graphics processor
  - Focus on parallel, multi-thread throughput, low resolution data (short words)

Characteristics

- Microprocessor:
  - Performs well on wide range of instruction/data mixes
  - Binary compatibility across product ranges
- Signal processor:
  - Performs well on scalar products
  - Weaker compatibility guarantees (if any)
- GPU:
  - Performs well on (certain) highly parallelizable codes
  - Compatibility similar to signal processors

Design cases

Embedded HW/SW design cases

1. Have hardware, need software
2. Have software, need hardware
3. Have clean paper

- Affects approach
  - Rarely <u>pure</u> case of any of these!

1. Software w/ given hardware

- Similar to other programming tasks
- …but in embedded programming, resource constraints require special care
  - Real-time response times
  - Limited memory (often w/o VM or memory protection!)
  - Power

EDA223

2. Hardware w/ given software

- Select (COTS) processor that fulfils requirements on…
  - Performance
  - Compatibility (application code, firmware, O/S)
  - Cost (NRE, production, licensing, etc)
- … or if absolutely necessary, develop (parts of) processor hardware

Performance requirements

- How assess processor or system performance?
  - Preferably, by running the relevant software on relevant hardware
- But hardware and/or software may not exist yet!
  - What to do?

Fake the software

- If target software not ready, use other software as placeholder
- Select something “similar” to target software
  - Earlier version of target s/w
  - Well-known benchmark suites
  - Purpose-written “benchmark” software
- Evaluate performance, power, etc. using the placeholder software

Fake the hardware

- If no hardware, use a simulator to provide…
  - (bit-true?) emulation of processor instruction set
  - … + memories and caches
  - … + peripherals …
- Example: www.gem5.org
- Trade off level of detail w/ execution time
  - 2–3 OoM slower than real hardware
- Real-time behavior not provable by simulation

Processor development

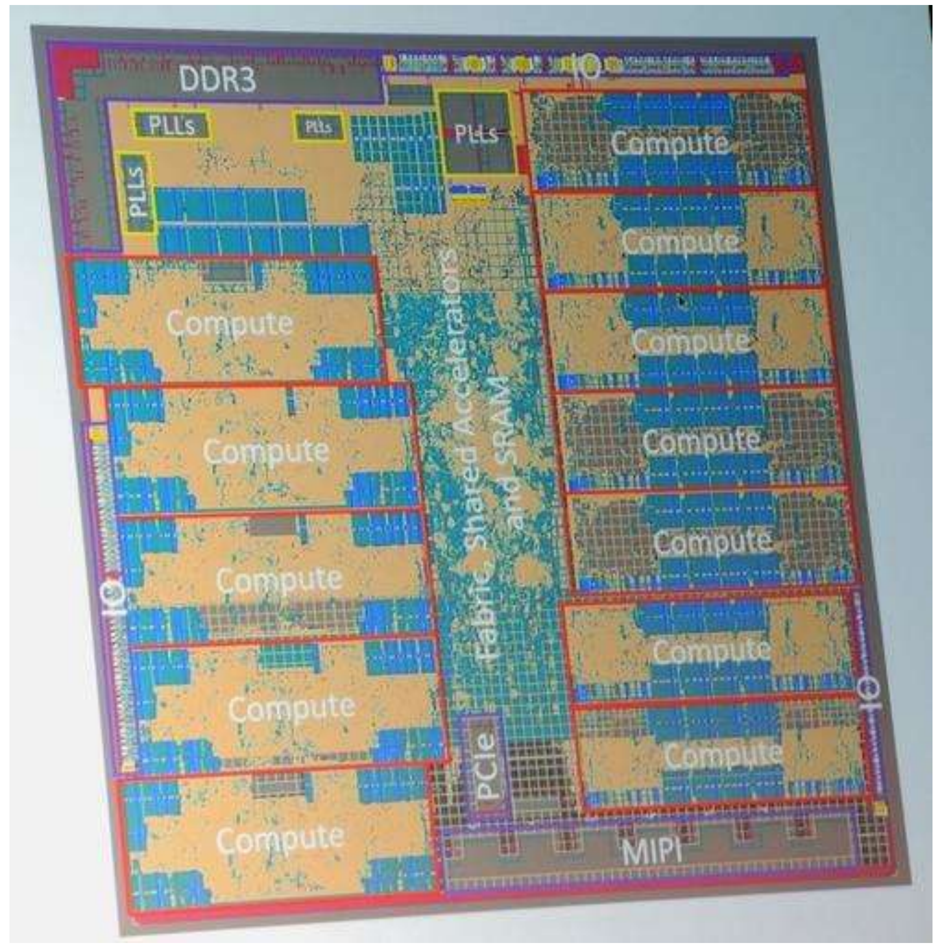
- Benchmark-driven
  - Select/create appropriate benchmark (suite)
- Application-driven (single application!)
  - Highly specific architecture possible
- Incremental approach (for either of the above)
  - Add few specific instructions to existing core, evaluate, repeat
  - Commercial tools available

Commercial example: Tensilica Xtensa

- Start from “bare-bones” processor
- Use performance evaluations to steer architecture modifications
  - Software tool chain support
- White paper uploaded (note: this is marketing material!)

[Tensilica is now part of Cadence.]

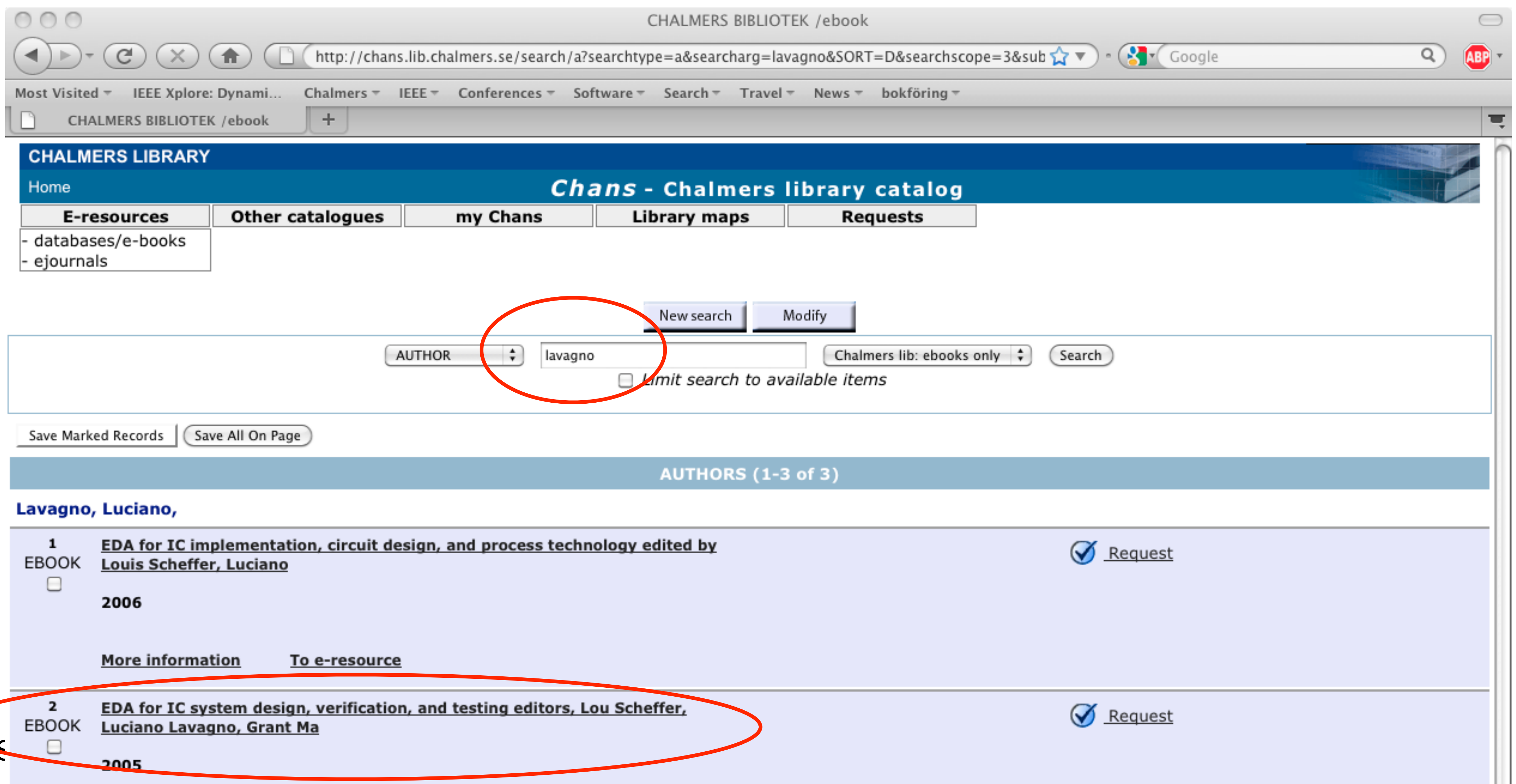
Microsoft HoloLens (2016)



- VR headset chipset
- 24 Tensilica cores (Intel Atom host)
- 28nm, 12 x 12 mm, 65M gates
- < 4W

Background reading

- See Lavagno et al: “EDA for IC system design, verification, and testing”, Ch. 10
  - Available in E-book form at Chalmers Library (chans.lib.chalmers.se)



3. Clean-paper start (rare)

- Need system-level specification
  - Executable specification preferable
- Separate into hw, sw parts
- Select processor(s), O/S, firmware
- Handle hw/sw interfaces
  - Lowest level: addressable registers
  - Higher level: O/S drivers

the hard part

Experience is invaluable!

Multiple processors

- Today, many (most?) embedded systems contain multiple processor cores

EDA284

- Full complexity of distributed systems and of parallel computer systems

TDA596

DAT280

Hardware vs software: compare and contrast

Is software soft?

- Term coined multiple times in 1950s (John Tukey, et al)
  - Contrast to computer hardware
- Later, often understood to signify ease-of-modification
  - Edit/compile/test vs. soldering
- Today, software may be <u>more</u> difficult to alter than hardware
  - Large and growing
  - Portability may require tests on many hardware combinations
  - Regression tests after each change

Hard? Soft?

- Reconfigurable hardware straddles boundary
  - Cf. lab sessions!
- Rapid development w/o the “processor” architecture
- Can avoid some bottlenecks
  - Adaptable wordlength, extreme parallelism, no instruction fetching

Example (non-embedded)

[Kevin Cushon]

- Simulation of novel error-correcting codes
  - BER of 1e–15 and below
- Software implementation does 550 data frames (~1e4 bits) per processor core per second
- 3600 datapaths on large FPGA does 10M frames per second
  - 18000x speedup
- On FPGA, ~1 frame error per hour at BER = 1e–15
  - 2 core-years in software

Processor-with-FPGA

- FPGA with (one or more) processor cores included on die
- Example: Xilinx “Extensible Processing Platform” (EPP)
  - ARM processor cores + FPGA fabric
- Ex: Zync-7000 series
  - Reading material uploaded (marketing!)

Processor-<u>in</u>-FPGA

- Define processor in HDL, implement on FPGA
- May make sense for same reason as processor in ASIC: hardware re-use, development flexibility
  - Adaptable to oddball requirements (e.g. wordlength)
  - Likely not competitive with “hard” processor – <u>unless</u> FPGA (w/ spare logic) already included in system!
- Xilinx “Picoblaze” Product Brief uploaded
  - Similar designs available from other FPGA manufacturers and as open source

C-to-FPGA

- Describe algorithms in “software” language, map directly to FPGA
- Examples: Catapult-C (offered by Mentor Graphics)
  - SystemC (C++ with class libraries and restrictions)
  - Data sheet uploaded (marketing!)
- Processor? Hardware? Software?

Alternatives

- Pure VHDL description
- Fixed-ISA processor
  - Separate package
  - Hard macro
  - Soft macro
- C-to-hardware (ex. Catapult)
- “Tweakable-ISA” processor
  - ASIC (ex. Tensilica)
  - FPGA (ex. Xilinx Picoblaze)
- FPGA w/ hard processor macro (ex. Xilinx Zync)
- …

HW vs SW

- Behavior is behavior.
    - HW or SW is an implementation detail
- Hardware/software border often fuzzy
    - New technologies try to combine benefits of both paradigms
    - A challenge to find practices that cover the divide
- “Cultural” differences persist!
    - Ask a hardware guy about software developers, etc.

Summary

- In practice, software components in almost all electronic-system design projects
    - Hardware/software border often fuzzy
- Design-alternative menagerie large and growing
    - Designers need to keep up!
    - Cf. “Technology platforms” lecture…
- “Agility” requirements push towards larger role for software