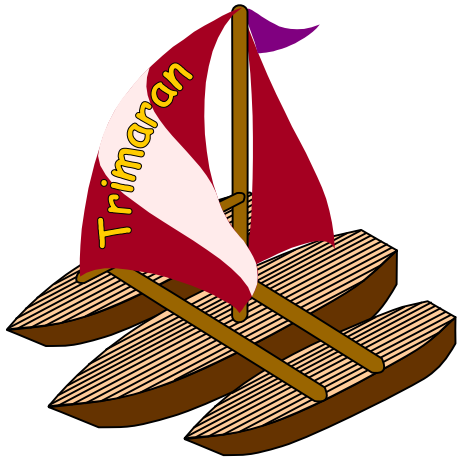
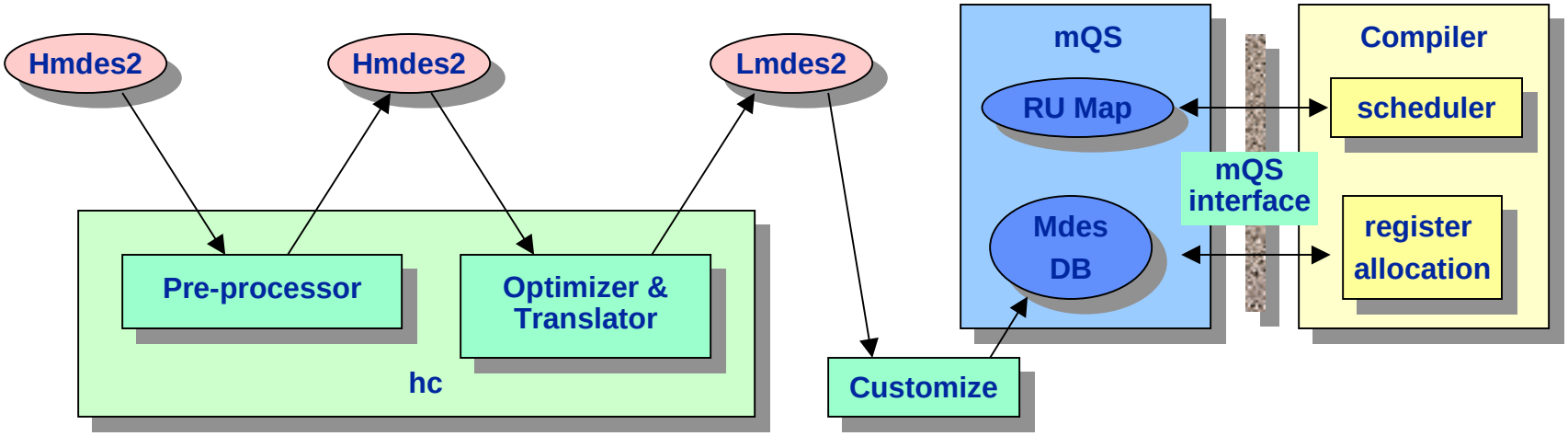


mdes Tutorial: Part I

History and Overview

- The goal: to minimize the number of assumptions built into the compiler back-end regarding the target machine



- The processor types have been served by mdes this far
  - Cydrome: Cydra 5 (VLIW, complex, non-parametric)
  - HPL: HPL-PD (VLIW, simplified, parametric)
  - IMPACT: products (superscalar, complex, non-parametric)

Outline of This Tutorial

- ILP code generation
  - new emphases on concepts
  - a model of code generation
- This information required for code generation
  - the code selector
  - the scheduler
  - the register allocator
- The mdes Query System (mQS) and its interface
- The Hmdes2 machine description language

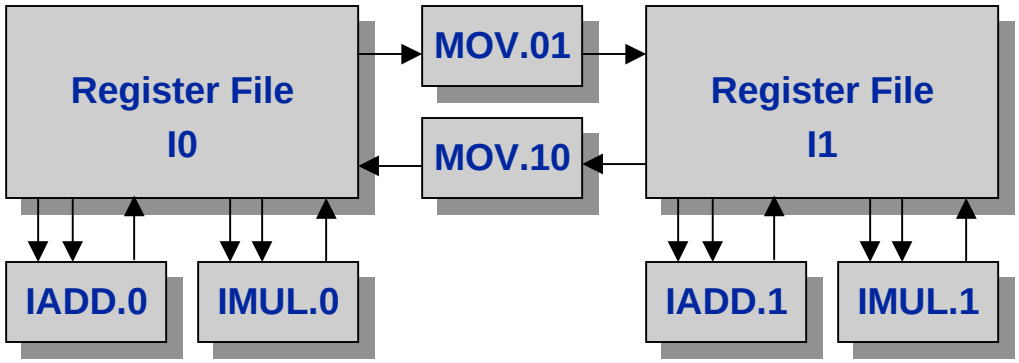
Operations

- An operation
  - an opcode
  - a list of register specifiers, one for each operand
- A (machine-independent) **semantic operation** is the output to the code selection module
  - a semantic opcode
  - a list of virtual register specifiers, one for each operand
- A (machine-specific) **architectural operation**
  - an architectural opcode
  - a list of architectural register specifiers, one for each operand
- An architectural operation is one that is specified in the Architecture Manual--these are the only operations that the hardware understands

Additional Challenges in ILP Code Generation

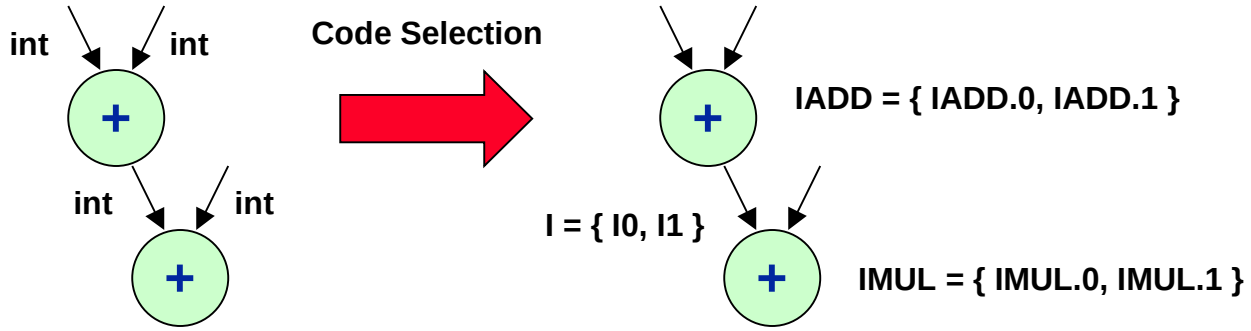
- In a conventional compiler, code selection consists of selecting the appropriate architectural operations to implement the semantic operations
- In an ILP processor there are multiple, machine-specific alternatives for how a semantic operation can be implemented
- In general, each architectural operation can correspond to multiple alternatives
- In an ILP compiler, code selection consists of selecting the appropriate alternatives to implement the semantic operations

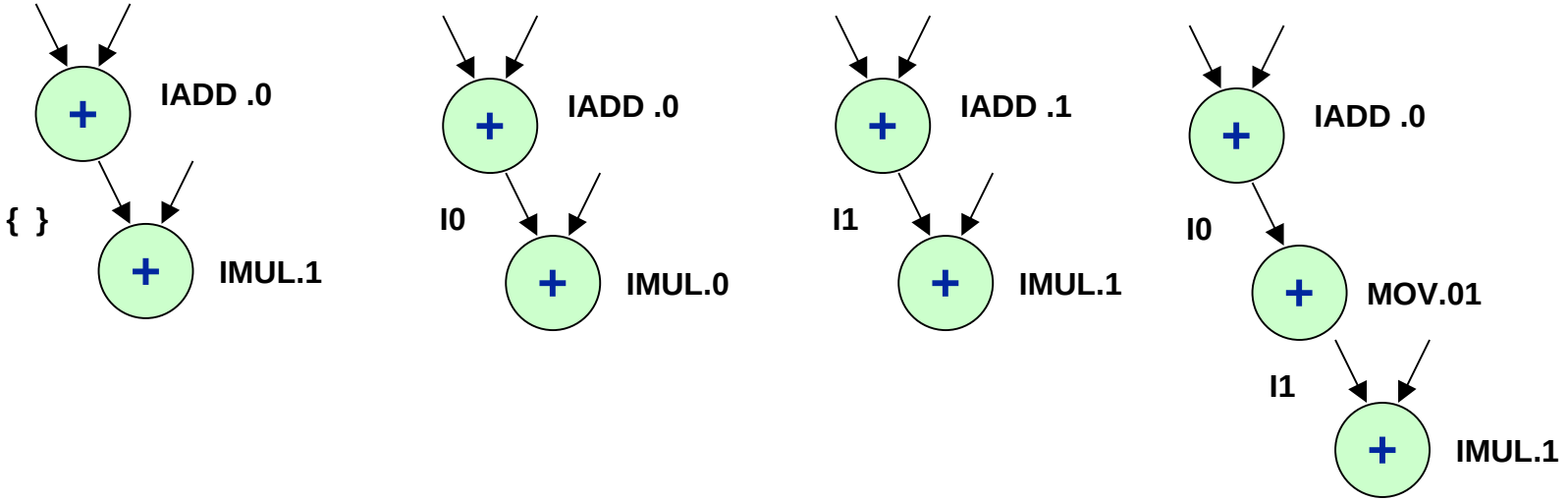
ILP Code Selection



Opcode	Description	I/O descriptor	Semantics
IADD.0	Integer Add	P? I0, I0 : I0	dest = src1 + src2
IADD.1	Integer Add	P? I1, I1 : I1	dest = src1 + src2
IMUL.0	Integer Add	P? I0, I0 : I0	dest = src1 x src2
IMUL.1	Integer Add	P? I1, I1, : I1	dest = src1 x src2
MOV.01	Move from I0 to I1	P? I0 : I1	dest = src
MOV.10	MOVE from I1 to I0	P? I1 : I0	dest = src

ILP Code Selection (contd.)





Elcor's compilation steps

- Conventional code selection
  - replaces **semantic** operations with (machine-specific) **generic** operations
- ILP code selection
  - binds generic operations to register-file-constrained operations
  - obviate the need to modify the graph while scheduling
  - ensure a non-empty intersection of the destination space of a producer operation and the source space of the consumer operation, based on the machine structure
- Scheduling
  - binds on an **alternative** and, thus, to an architectural opcode

Elcor's Compilation Steps (contd.)

- Register allocation
  - binds each operand to a specific architectural register
- Scheduling

Abstract Compiler Operations

- An alternative

an alternative =

an **architectural opcode** (with associated semantics) a set of architectural register file binds as specified by an **I/O descriptor**

with

resource usage as specified by a **reservation table**
latencies as specified by **latency descriptor**

- A **fully qualified operation** corresponds to a specific alternative in which all of the operands have been bound to specific architectural registers
- A **generic operation** specifies the set of all alternatives that have the same semantic behavior

Abstract Compiler Operations (contd.)

- The compiler works with a hierarchy of abstract **compiler operations** which are incrementally bound to a fully-qualified operations

Information Required By The Compiler

- ILP code selection (determination of operand constraints with respect to the set of legal register files that may house the operands
  - **I/O descriptor**: source / destination register file constraints for operations
  - Register file: the set of compatible register types
- Edge drawing
  - **Register**: overlapping registers, i.e., that have at least one bit in common
- Edge delays
  - **Latency descriptor**: source sampling / result update times for operations

Information Required By The Compiler (contd.)

- Legality of scheduling at a given time with respect to resource conflicts
  - **Reservation table**: resource usage over time for each operation
- Deferred code selection
  - **Operation descriptor**: the architectural opcode corresponding to a full-qualified operation
- Lifetime calculation
  - **Latency descriptor**: register allocation and deallocation times
- Register allocation options
  - **Register file**: the set of legal registers for allocation

Register and Register Files

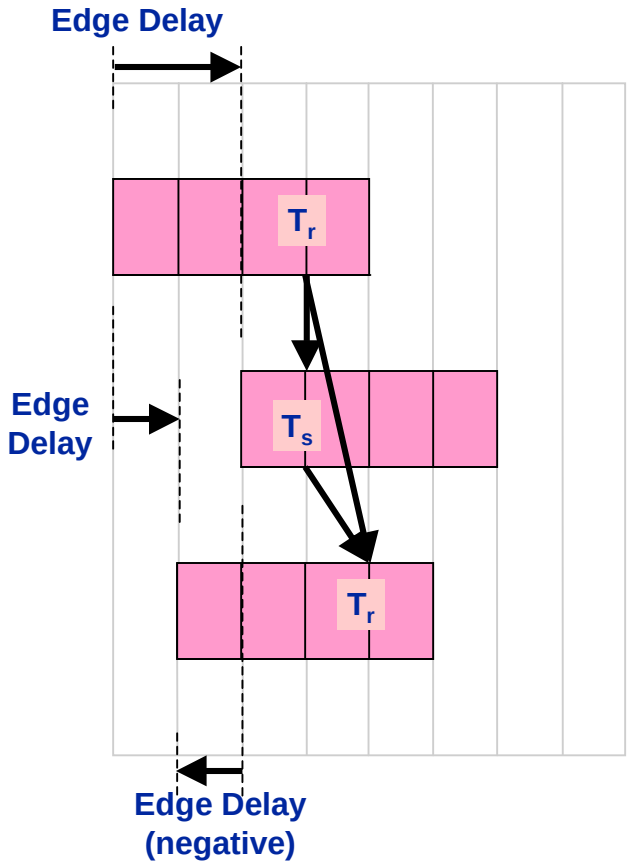
- Register
  - identifier
  - the set of overlapping registers
- Register file
  - a set of interchangeable static registers, and/or
  - a set of interchangeable rotating registers, and/or
  - a set of literal (immediate) values
  - a set of register types that are compatible with these registers
  - currently, the set of real register files is constrained, but the number of literal register files is unlimited

Operation formats (I/O descriptors)

- Field type (I/O set)
  - A set of register files (that can be the source/destination for a given operand of a given operation
- Operation format (I/O descriptor)
  - The specification of the field type (I/O set) for each operand (predicate, source, destination) for an operation or set of operations
- Sample I/O descriptors

Opcode	Description	I/O descriptor	Result Specifications
ADD	Integer Add	P? F,F : F	dest = src1 + src2
MOVFI	Move from F to I	P? F : I	dest = src
IADD	Integer Add	P? I L1, I L1 : L	dest = src1 + src2
IMUL	Integer Multiply	P? I L1, I L1 : I	dest = src1 x src2

Edge Delays With Differential Latencies



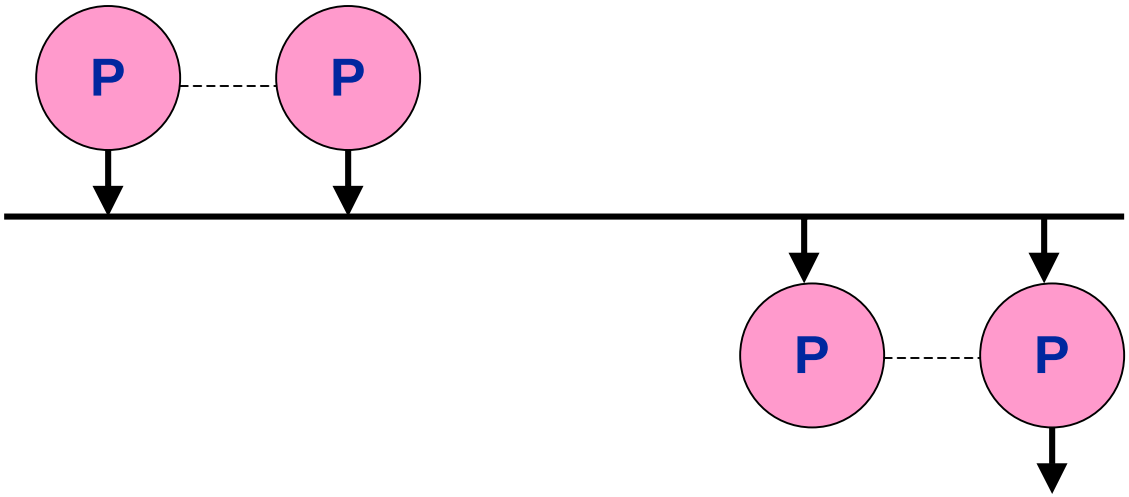
The edge delay specifies the number of cycles between the initiation of the operations at the two ends of the dependence edge

Computation of Edge Delays

The edge delay specifies the number of cycles between the initiation of the operations at the two ends of the dependence edge

Type of dependence	Edge Delay	Conservative Delay
Flow dependence	$T_r(P) - T_s(S)$	$T_r(P)$
Anti-dependence	$1 + T_s(P) - T_r(S)$	0
Output dependence	$1 + T_r(P) - T_r(S)$	$T_r(P)$

Computation of Lifetimes



Birth time	$\min[\ t(P_i) + T_a(P_i)\ ]$, for all i
Death time	$\max[\ t(S_j) + \max[T_s(S_j) + \max[T_s(S_j), T_x(S_j)\ ]$, for all j

T(X) is the time at which operation X is scheduled

Latency descriptors

Multadd (r1, r2, s1, s2, s3) = **r1 ← s1 x s2**
r2 ← r1 + s3

	Ts Input sample time (earliest) Earliest input deallocation time	Tr Result availability time (latest) Latest output allocation time	Tx Exception time (time) Latest input deallocation time	Ts Reservation time (earliest) Earliest output allocation time
s1	0		4	
s1	0		4	
s1	2		4	
s1		3		0
s1		4		0

All **differential latencies** are relative to the initiation of the operation

Reservation Tables

Relative Time	InstField02	InputBUS01	InputBus02	FPDivide_0	FPDivide_1	ResutBus01
0	X	X	X	X		
1				X		
2				X	X	
3					X	
4					X	X

- A resource is anything (pipeline stage, bus, instruction field) for which it is possible to have a collision
- Reservation tables automatically exclude all forbidden scheduling intervals

Note: the notion of "executing on a functional unit" is imprecise