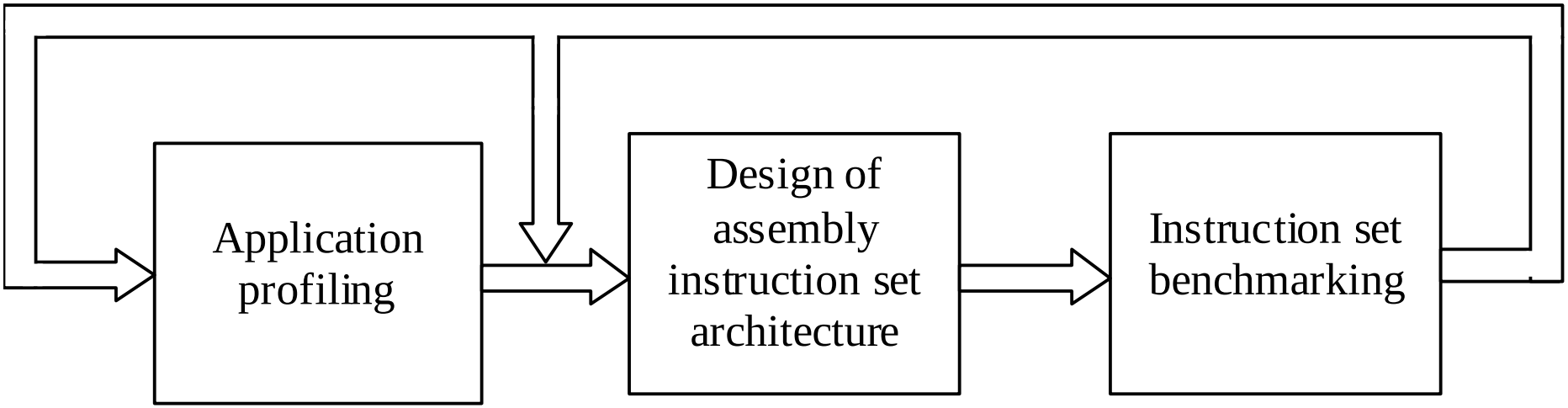
Chapter 6

Profiling

Simplified assembly instruction set design flow



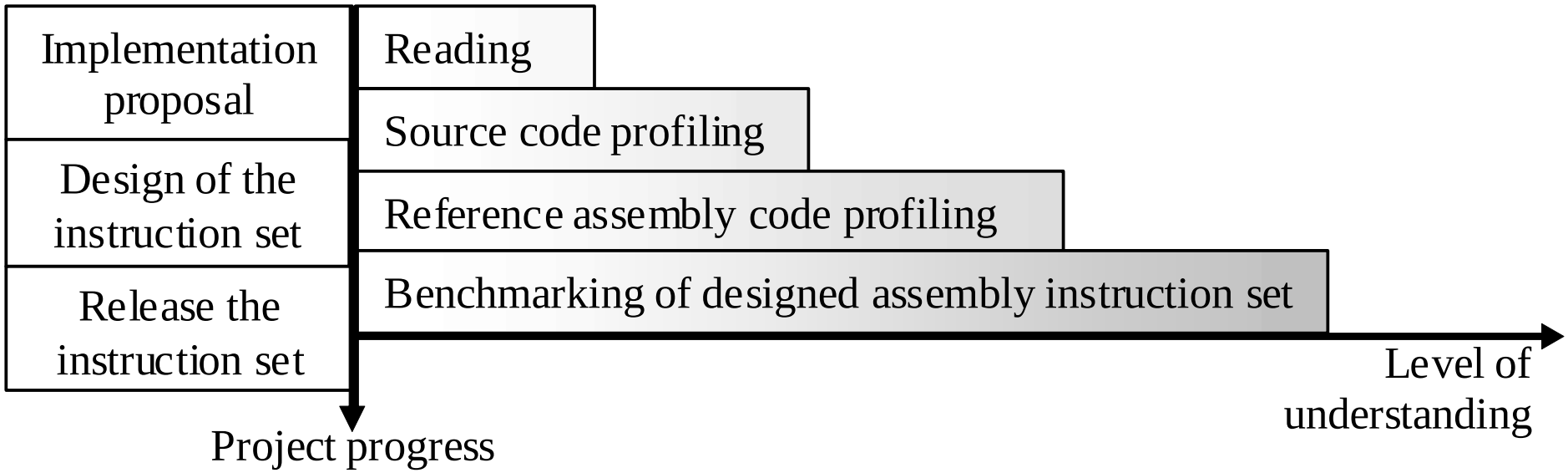
Contents

- What is profiling and why profiling
- Static profiling
  - Coarse profiling and fine profiling
- Dynamic profiling
  - Instrumentation and dynamic profiling
- Evaluate profiling quality

ASIP design: start from applications

- Divided ASIP design into two steps:
  - 1: understand the functions of the application and its requirements on performance.
  - 2: find ways to minimize costs while meeting requirements on flexibility and performance.
- Code profiling analyzes the application code structure and measures its execution time and memory cost.

Ways to understand applications



Why profiling?

- Algorithm computing cost
- Memory access cost
- The cost of running program flow control
- The execution cost for hardware and timing resource management
- Cost to serve asynchronous tasks
- Synchronization and communication as well other cost of parallelization

What to profile

- Algorithm computing cost
- Memory access cost
- The cost of running program flow control
- The execution cost for hardware and timing resource management
- Cost to serve asynchronous tasks
- Synchronization and communication as well other cost of parallelization

To support instruction set design, profiling should identify:

- The most MIPS consuming functions that should be accelerated
- The most frequently appearing functions that should be accelerated
- Datapath architecture satisfying questions 1 and 2
- Memory and bus architecture supporting questions 1 and 2 with the minimum on-chip memory cost

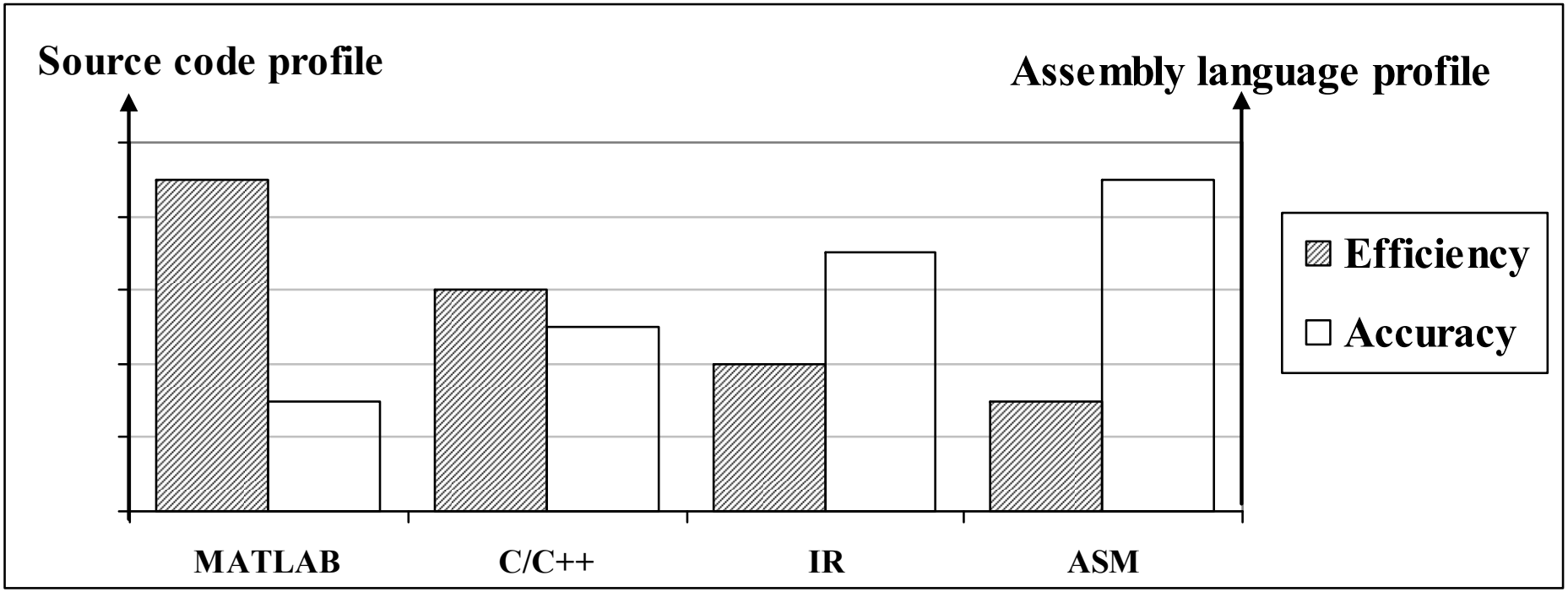
Static and dynamic profiling

- Static profiling is given by analyzing the source code instead of running it
  - WCET analysis
- Dynamic profiling is performed by executing the source program and accumulating the execution time

Profiling process

1. Decide the scope of a profiler: not easy
2. Analyze source code (static profiling).
3. Prepare and configure the profiling tool (instrumentation).
4. Run dynamic profiling.
5. Analyze the results of static/dynamic profiling.

Different kinds of profiling



Dynamic profiling at IR level is possible if IR simulator is available
Memory accesses might be excessively counted

Static profiling in 4 steps

1. Generate CFG (Control Flow Graph), identify the basic blocks are the lowest component.
2. Perform cost annotation on each branch of the CFG.
3. Identify cost accumulation based on annotations on all branches of the CFG.
4. Analyze the result and find the cost of the true critical path.

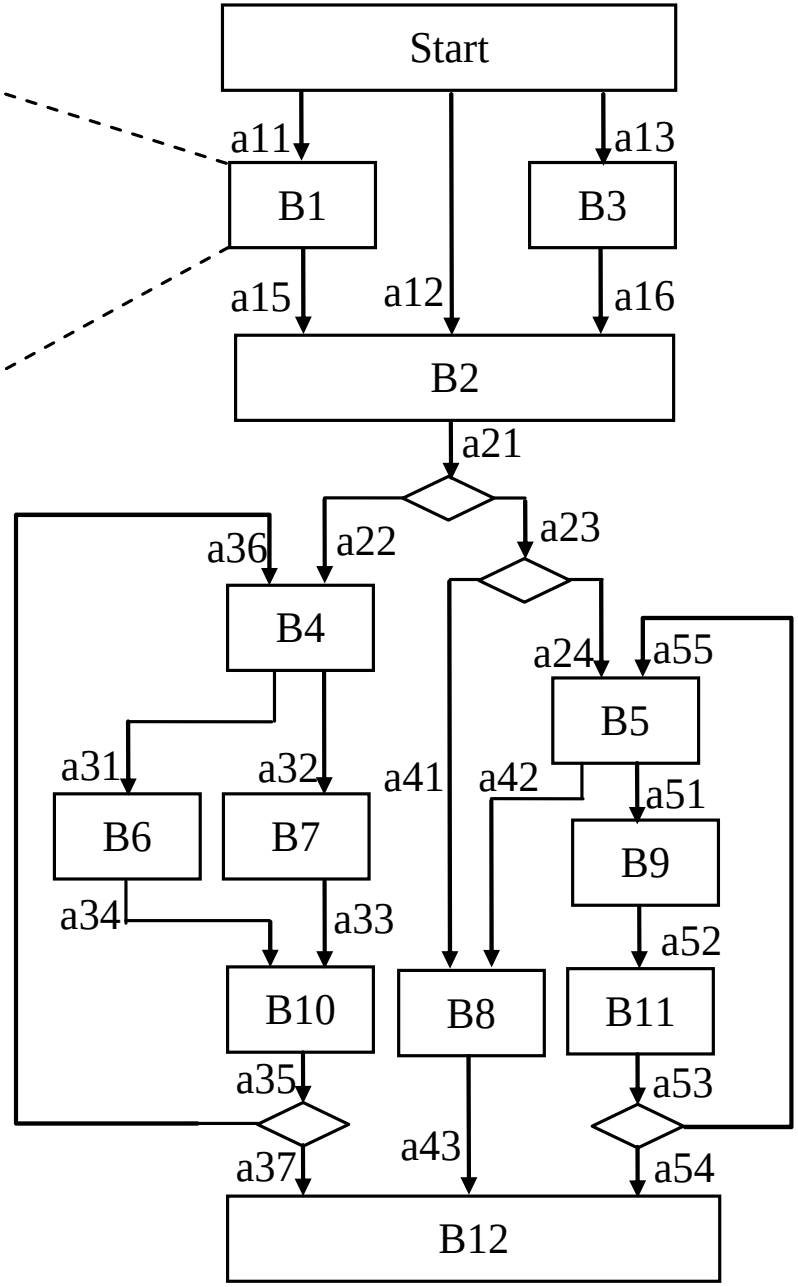
Fine-grained Static Profiling

- For instruction set design, expose computation cost in a BB (Basic blocks)
- Counting operations in a BB including
  - +, −, *, /, and **%**.
  - logic operations **&&**, ||, and **!**.
  - Bitwise operations ~, ^ , **&**, | , << , and >>
  - Relation operators >, >=, < ,<=, ==, and **!=**
- Memory accesses and communication should be also profiled

An example

Basic block 1:
10 MAC
20 ADD / SUB
15 scaling (shift)

Fine-grained profiling



Following the example

Coarse-grained profiling		Fine-grained profiling		
Basic block	Total	Arithmetic operations	Logic / Shift	MAC
B1	45	20	15	10
B2	12	10	2	0
B3	3	2	1	0
B4	140	10	30	100
B5	8	4	2	2
B6	145	25	0	120
B7	122	2	10	110
B8	53	50	3	0
B9	14	10	2	2
B10	214	12	2	200
B11	11	8	1	2
B12	6	5	1	0

Load/store and move

- Depend very much on the target architecture
  - Instruction set
  - Register file
  - Data memory
- Architecture independent load/store
  - Input variables of a procedure
  - Output of the procedure
  - Changed global variables
- Hidden load/store: spilled vatiables

Coarse-grained Static Profiling

- Coarse-grained: for architecture design
  - fine-grained: for instruction set design
- Identify
  - Time-consuming BBs
  - Frequently executed BBs
  - BBs that use large part of program memory
  - BBs that use large part of data memory

An example: IEEE802.11a/g: appearance of BB × cost of BB

Tasks	Algorithms	MOPS	Data types
Viterbi decoding	Viterbi decoding algorithm	1700	Short integer
Receiving Filter	Complex FIR convolution	960	Complex
De-interleaving	Permutation algorithm	260	Bit
Track fast fading	FFT and vector diff	160	Complex
Fading compensation	F-domain LMS	200	Complex
Sampling phase offset	FFT / phase decision	100	Complex
Demapping	Decision FSM	80	Integer
Descrambling	Polynomial algorithm	80	Bit
Packet detection	Autocorrelation	80	Complex
Energy detection	Autocorrelation	80	Complex
IFFT/FFT	64 points FFT and IFFT	60	Complex
Payload Rotor	Vector product	40	Complex
Normalization	1/x and vector product	40	Complex
TOTAL		3840	

Instrumentation for Coarse-grained dynamic profiling

- The purpose of instrumentation of a coarse-grained profiling is to understand the software system and the structure of the code at a higher level.
- It identifies the critical paths, the scope of the algorithms, the general execution behaviors of kernel algorithms, and the use of hardware in general.

Instrumentation for Coarse-grained dynamic profiling

- Probes or instrumentations are inserted to:
  - Check the frequency to reach each program mode (behavior of the high-level CFG).
  - Check the cycle cost of each basic block.
  - Check the frequency of invocation of a basic block (low level paths).

Instrumentation for Fine-grained dynamic profiling

- To get the knowledge of appearance of instructions inside the basic blocks
- To help decide:
  - Which appears frequently
  - Which should be accelerated

Example: Fine-grained dynamic profiling

Standards	G.723 6.3kb/s	
Operations in a voice frame	Dynamic	Static
16 bits Fractional MAC	222373	264810
16 bits Integer MAC	58565	74768
32 bits arithmetic left shift	8550	10242
16 bits absolute value	6975	8945
32 bits absolute value	6937	9068
16 bits multiplication with 16 bits rounded result	3308	3478
32 bits arithmetic right shift	2625	4075
16 bits multiplication 32 bits result	2126	7076
32 bits subtraction	638	841
32 bits addition	557	682
32 bits normalization	378	782
32 bits by 16 bits multiplication	339	1406
16 bits multiplication 16 bits result	329	7544
TOTAL main arithmetic operations in a frame	313772	393968
The frame length (milliseconds)	30	30
TOTAL main arithmetic operations per voice sample	1307	1642
TOTAL arithmetic MOPS	10.46	13.14

Hidden cost: Shall be estimated

- To prepare and terminate a subroutine call
- The cost of preparing and running HW (DMA and other IO transactions)
- The cost of the top-level management of a program
- The cost of thread and interrupt handling
- The extra cost of parallelization (synchronization, communication, etc.)

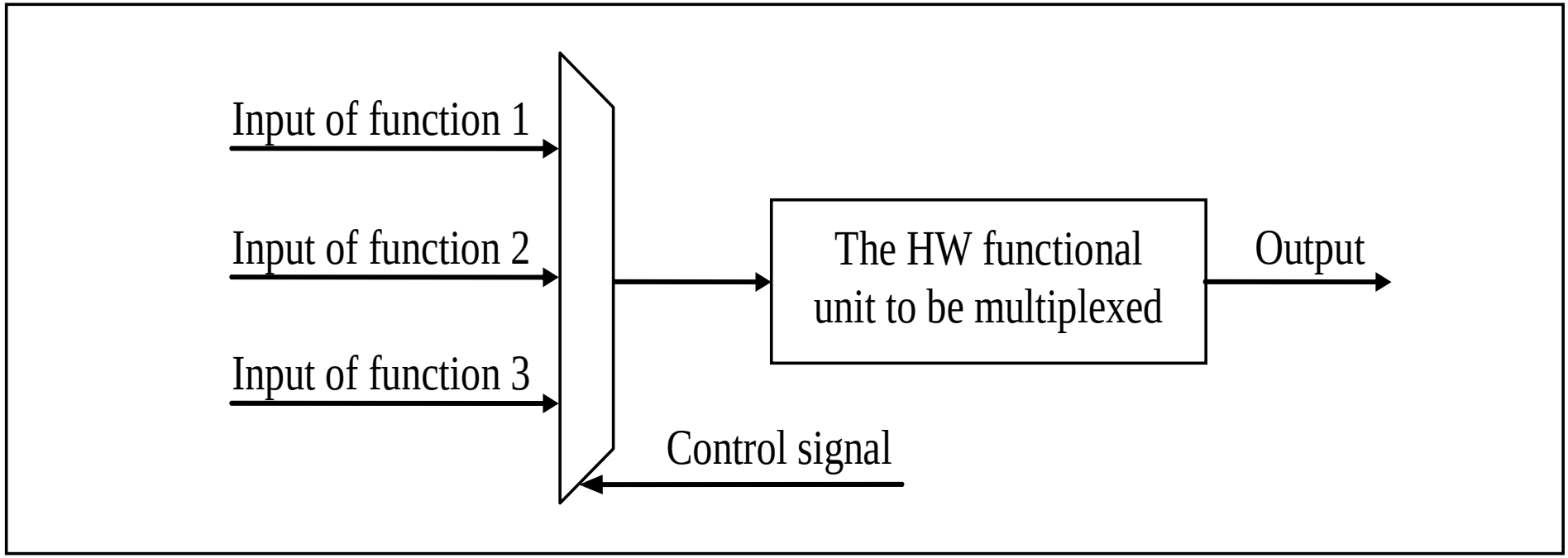
Hidden cost: Shall be estimated

- Hidden computing costs cannot be exposed before analyzing assembly level codes.
- When HWMR (HW multiplexing ratio) is low, the hidden cost could be a big trouble (?)
- Check the example: Running 802.11a and Bluetooth in parallel in the book.
  - Bluetooth data packet processing : 1 microsecond
  - WLAN 802.11a processing: 4 microseconds
  - EDF scheduling for two tasks (250MHz)
  - Thread switching: 50 cycles
  - 100/(250x4) = 10%

HW multiplexing

- A technique to share a hardware device or a functional unit (a full adder, a multiplier, a register file, etc.) for different purposes at different times.
- Hardware multiplexing can be implemented either by software or by configuring the hardware
- HWMR:
  - machine clock rate / signal sampling rate (?)

HW multiplexing



Review

- The quantitative design of an assembly instruction set requires deep understanding of applications.
- Profiling is a way to understand applications.

- Static profiling: Code analysis

- Dynamic profiling: Running code using carefully selected stimuli.