УДК 004.318

Байда Ю.В.
Московский физико-технический институт

Выбор оптимального варианта реализации кэша трансляции целевых адресов косвенных переходов для двоично-транслирующих систем

Системы двоичной трансляции находят широкое применение в современных вычислительных системах. Двоичная трансляция является одним из наиболее эффективных способов обеспечения совместимости вновь разрабатываемых вычислительных архитектур с уже накопленным объёмом программного обеспечения [1].

Одной из наиболее серьёзных причин, приводящих к потере производительности в системах двоичной трансляции, являются команды косвенной передачи управления (*англ.* indirect branch, indirect jump) [2]. Такие команды получают целевой адрес из регистра или памяти, содержимое которых может меняться во время исполнения программы. Поскольку этот адрес принадлежит исходному адресному пространству, его необходимо транслировать каждый раз, когда команда косвенного перехода поступает на исполнение, а не только при её первой трансляции.

Одним из способов уменьшения потерь, связанных с обработкой команд косвенной передачи управления, является включение в процессор аппаратной поддержки в виде так называемого кэша трансляции целевых адресов косвенных переходов (*англ.* indirect branch target address translation cache) [3]. Данное устройство представляет собой ассоциативную память, расположенную в ядре процессора, в которой хранятся пары адресов <*исходный адрес, оттранслированный адрес*>. Доступ к кэшу осуществляется с помощью специальных команд записи вида *puttc* r_{d1}, r_{d2}, которая помещает в кэш пару адресов <*исходный адрес, оттранслированный адрес*>, хранящихся в регистрах r_{d1} *и* r_{d2} соответственно, и чтения вида *gettc* r_d, r_s, которая по исходному адресу, расположенному в регистре r_s, возвращает соответствующий оттранслированный адрес в регистр r_d.

Размер, ассоциативность и политика замещения могут быть различными. Для определения оптимального варианта реализации кэша трансляции целевых адресов косвенных переходов в данной работе было проведено моделирование с помощью

системы инструментирования Pin [4] и тестов из пакета SPECint 2000.

В качестве основной модели в данной работе был выбран вариант реализации кэша трансляции целевых адресов косвенных переходов, при котором и для возвратов из функций, и остальных косвенных переходов используется общий кэш. При этом для возвратов из функций использовался механизм предзагрузки (после каждого вызова функции записывать в кэш трансляции целевых адресов косвенных переходов уже известный оттранслированного адреса возврата, тогда с высокой вероятностью он не успеет вытесниться из кэша и обращение в кэш за адресом возврата будет успешным).

Далее, для определения оптимального варианта реализации кэша трансляции целевых адресов косвенных переходов, был произведен выбор компромисса между производительностью и площадью, занимаемой устройством на кристалле. Оценка площади выполнена для технологии КМОП с технологическими нормами 45 нм.

На рис. 1 показана полученная зависимость между долей неудач при обращении в кэш трансляции целевых адресов косвенных переходов с предзагрузкой для возвратов из функций (стратегия замещения — LRU) и площадью устройства. На рисунке явно прослеживается зона оптимальных значений (отмечены на рисунке сносками).

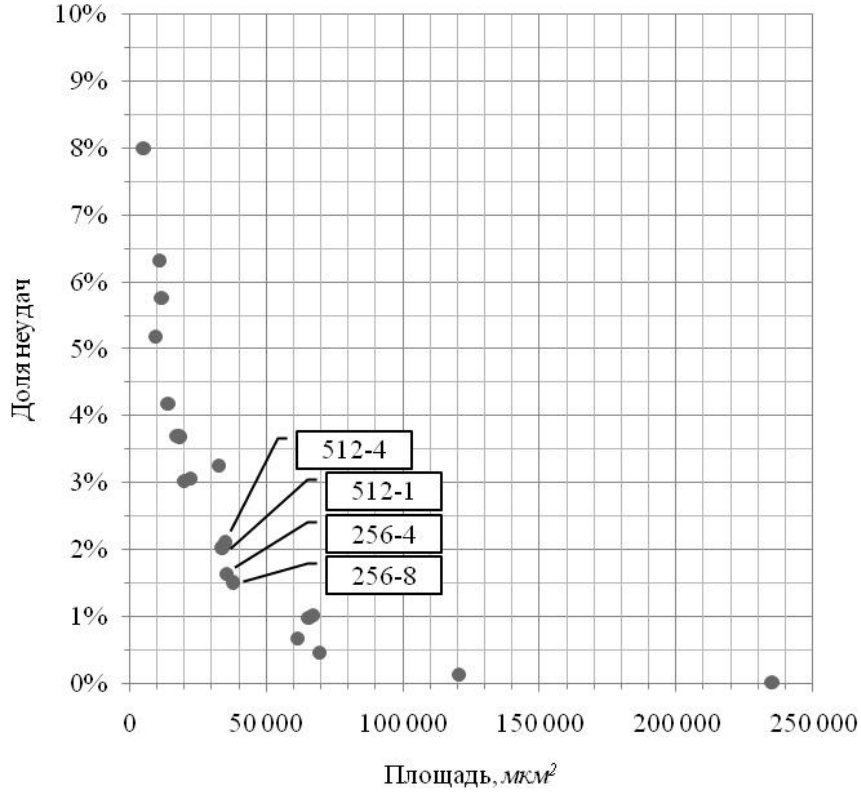

Рис. 1. Зависимость между долей неудач при обращении в кэш трансляции целевых адресов косвенных переходов с предзагрузкой для возвратов из функций и площадью устройства (стратегия замещения — LRU)

Из оставшихся вариантов в качестве наиболее оптимального был выбран следующий вариант: размер — 256 записей, ассоциативность — 4.

СПИСОК ЛИТЕРАТУРЫ

1. *Ермолович А.В.* Применение базы кодов в динамической двоично-транслирующей системе // Высокопроизводительные вычислительные системы и микропроцессоры: сб. научно-техн. тр. — 2003. — вып. 4. — С. 55–66.
2. *Hiser J. D., Williams D., Hu W., et al.* Evaluating indirect branch handling mechanisms in software dynamic translation systems // Proceedings of the International Symposium on Code Generation and Optimization. — 2007. — P. 61–73.
3. *Gschwind M.* Method and apparatus for determining branch addresses in programs generated by binary translation // IBM Technical Disclosure YOR8-1998-0334. — 1998.
4. *Luk C., Cohn R., Muth R., et al.* Pin: building customized program analysis tools with dynamic instrumentation // Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation. — 2005. — P. 190–200