소형 임베디드 프로그램의 실행 속도와 특성분석

정새암, 이정수, 김준성
중앙대학교 전자전기공학부
e-mail : {auxo,xmxm2718}@wm.cau.ac.kr, junkim@cau.ac.kr

Chacterization of Small Embedded Programs

SaeAm Chung, JongSu Yi, JunSeong Kim
School of Electrical and Electronics Engineering
Chung-Ang University

Abstract

In this paper, we analyze the characterization of Mibench, an embedded system benchmark program, using simplescalar simulator. The experimental results show Mibench generally is formed by lots of integer and memory access instructions. Especially, IPC of rijndael decoding is effected by cache size largely, but IPC of CRC32 is few effected by cache size or branch predicting algorithm.

I. 서론

오늘날 우리는 임베디드 시스템의 세계에서 살고 있다고 할 수 있다. 전자 계산, 일정 관리, 휴대 통신, 웹 서핑, 동영상 플레이 등의 수많은 작업들을 한 가지 시스템으로 모두 처리하기도 하지만, 많은 시스템들은 특수한 한두 가지 목적을 위하여 제한된 자원을 가지고 시스템을 구현한다[1]. 이러한 특수한 목적을 지닌 시스템들에 적합한 프로세서들이 두드러지는 추세이다. 사회 각 분야에 걸쳐 다양한 목적의 임베디드 시스템이 구현되어 사용되고 있다. 이들은 그 특수한 목적에 따라 각기 다른 종류의 기능을 수행한다.

이러한 임베디드 시스템들을 목적에 맞게 선택하기 위해서는 각 시스템의 특성을 잘 판단해야 한다. 그리고 각 시스템에 적합한 프로그램을 구현하여야 하며, 이는 인스트럭션의 구성과 밀접한 관계가 있다. 본 논문은 Mibench 프로그램을 대상으로 임베디드 프로그램의 인스트럭션 구성과 그 특성을 분석한다.

II. 본론

Mibench는 임베디드 시스템의 성능 측정을 위해 미시간 대학에서 만든 벤치마크 프로그램이다[5]. 이 벤치마크 프로그램은 크게 automotive, consumer, network, office, security, telecomm의 6개 그룹으로 구성되어 있으며 이 각각은 산술 알고리즘, 멀티미디어 인코딩/디코딩, 라우팅 알고리즘, 문서 작성, 보안, 통신 관련 인코딩/디코딩 프로그램 등으로 구성되어 있다. 이 각각의 그룹은 많은 임베디드 시스템들의 전반적인 기능을 두루 벤치마킹 해주고 있다. automotive 그룹은 6개의 서브 프로그램으로 구성되어 있으며, consumer 그룹은 9개, network 그룹은 2개, security 그룹은 7개, telecomm 그룹은 7개의 서브 프로그램으로 구성되어 있다.

임베디드 시스템의 경우 그 목적에 따라 다른 여러 특성을 보여주기 때문에, 이러한 목적들을 지닌 벤치마크 프로그램으로 그 기능에 대한 성능 측정이 필요하다. 또한 벤치마크 프로그램을 이용하기 위해선, 벤치마크 프로그램들이 지닌 특성을 이해하는 것이 중요

하다.

실험은 SimpleScalar 3.0[3] 을 이용한 환경에서 진행되었다. SimpleScalar는 Alpha, PISA(Portable Instruction Set Architecture), ARM, x86 인스트럭션 셋을 에뮬레이팅 해 주는 시뮬레이터로, 이 실험은 Mibench의 automotive, consumer, network, security, telecomm의 5 가지 그룹을 PISA instruction set architecture 로 포팅하고 small 데이터 셋을 사용하였다[2][3]. L1 캐시는 instruction 캐시 16kb, data 캐시 16kb를 사용하였고, L2캐시는 256kb, 분기 예측기는 bimodal 분기 예측기를 기본 옵션으로 사용하였다.

표1 mibrench 그룹별 instruction number

automotive	inst num
basicmath	177835342
bitcount	99747546
qsort	43755883
susancorners	2036047
susanedge	4908402
susansmooth	61544039

security	inst num
blowfishde	209522
blowfishen	209518
rijndaelde	37543577
rijndaelen	37602828
sha	35262279

consumer	inst num
jpegde	7340605
jpegen	27266535
lame	491945436
median	177829020
tiff2bw	52095565
tiff2rgba	45711000
tiffdither	295283297

telecomm	inst num
adpcmc	30212703
adpcmd	25244736
CRC32	27448804
FFT	92032276
FFTinv	136853638
gsmde	11388262
gsmen	34435592

network	inst num
dijkstra	94957377
patricia	138005743

표2 mibench 그룹별 인스트럭션 분포(단위: %)

	load	store	uncond	cond	int	float
automotive	33.55	16.80	3.80	6.52	37.16	2.17
consumer	21.78	9.72	1.44	9.47	54.08	3.51
network	22.70	8.50	5.35	11.53	50.84	1.10
security	25.20	6.61	1.08	3.04	64.06	0.00
telecomm	13.23	5.90	4.18	14.70	61.33	0.66

표 1과 표 2는 Mibench의 5가지 그룹의 인스트럭션의 분포를 보여주고 있다. 각 데이터들은 프로그램의 크기가 모두 다르기 때문에 절대치 대신 비율 데이터를 취하였으며, 동일한 내용에서의 증가율이 아니기 때문에 산술평균을 취한 값을 취하였다[4]. 결과 값을 크게 인스트럭션의 수, 총 소요된 클럭 수, 메모리 연산, 정수 연산, 실수 연산, 컨트롤 연산의 비율로 비교해 보았다. 전반적으로 메모리 접근 인스트럭션과 정수 연산 인스트럭션들이 주로 사용되는 것을 알 수 있으며. 부동소수점 연산의 사용이 적은 것을 볼 수 있다. 전통적인 산술 알고리즘쪽 구성인 automotive 그룹은 보다 많은 메모리 접근 연산을 하고 있으며 처음 예상과는 다르게 산술 연산의 비중은 상대적으로 낮았다. 반면, 통신과 관련된 인코딩과 디코딩 프로그램으로 구성되어 있는 telecomm 그룹에은 상대적으로 매우 적은 메모리 접근 연산을 보이고 있다.

표 3은 mibench의 그룹별 IPC와 분기예측 실패율과 캐시 미스율을 보여주고 있다. telecomm 그룹은 많은

표3 mibench 그룹별 IPC/IPB/miss rate

	IPC	IPB	bp miss	il1 miss	dl1 miss
automotive	1.58	14.01	6.17%	1.46%	0.99%
consumer	1.86	13.71	4.23%	0.19%	1.42%
network	1.49	5.96	4.16%	3.86%	0.36%
security	1.80	30.27	3.59%	1.86%	0.59%
telecomm	1.62	7.01	7.14%	1.39%	0.06%

분기문과 분기 예측 실패율에도 불구하고 위 각 그룹들에서 평균 정도의 IPC를 보여주고 있다. network 그룹의 경우 평균적인 메모리 연산을 지니고 있으나, 분기문이 차지하는 비율이 크기 때문에, 분기 예측 실패 비율이 적더라도 인스트럭션 캐시 미스가 많아져 전반적인 속도가 저하됨을 알 수 있다. 그리고 security 그룹의 경우 다른 그룹들에 비해 훨씬 분기가 적기 때문에 비슷한 캐시 미스를 보여주는 다른 그룹들에 비해 IPC가 큼을 알 수 있다.

두드러지는 세부 특징으로 security 그룹의 rijndael 프로그램의 decoding 부분이 다른 프로그램과 달리 캐시 메모리의 크기에 영향을 크게 받는(IPC 1.30 -> 2.52) 것과 network 그룹의 CRC32 프로그램의 IPC가 캐시의 크기나, 분기 예측 성능과 무관한 IPC값을 보였다는 것이다. 이는 CRC32 프로그램은 심플스칼라에서 비교적 많은 시간 동안 스트리밍 모드로 동작함을 유추할 수 있었다.

Ⅲ. 결론 및 향후 연구 방향

Mibench의 각 프로그램들은 각기 다른 인스트럭션의 구성으로 되어 있지만, 그룹별로 관측했을 때, 일부 공통점과, 그 공통점의 발생 원인을 알 수 있었다. 향후 연구는 Mibench가 시뮬레이션 되는 동안 Simple Scalar의 파이프라인이 어떠한 상황에서 멈추게 되고, 스트림 모드로 진행될 수 없는지 분석하며, 그 특성을 보다 명확하게 관찰하여 임베디드 시스템 성능 최적화를 위한 기반 자료를 제공하는데 그 초점을 맞출 예정이다.

참고문헌

[1] 오선진, 임베디드시스템 소프트웨어개발방법론, 한올출판사, 2007

[2] Doug Burger, Todd M. Austin, The SimpleScalar Tool Set, Ver 2.0, 1997

[3] SimpleScalar LLC, ANNOUNCE-3.0, 2003

[4] David J. Lilja, Measuring Computer Performance, Cambridge University Press, 2000

[5] http://www.eecs.umich.edu/mibench/