Alpha CPU의 Instruciton과 GAS의 사용법

삼성 소프트웨어 멤버쉽

리눅스 사용자 모임

정 병 수

1998년 11월 24일 초본

목 차

제 1 절 Alpha Instruction의 분류

1.1 Alpha Register의 종류

1. Program Counter : 1개 PC
2. Integer Registers : 32개 R0 $\sim$ R31
3. Floating-Point Registers :32개 F0 $\sim$ F31
4. Lock Registers : 2개
5. Processor Cycle Counter (PCC) Register : 1개
6. Operation Registers :
   (a) Memory Prefetch Registers
   (b) VAX Compatibility Register

1.2 Alpha Instruction의 종류

1. Memory (Mem) : 메모리에 관한 명령

   Memory/function code (Mfc) : 메모리에서의 함수 한 명령

   Memory (Mem) : 메모리에 관한 명령
2. Branch (Bra) : 분기 명령
3. Floating-Point (F-P) : 실수 명령
4. Operate (Opr) : 일반적인 명령
5. PALcode (Pcd) : PALcode에 관한 명령

1.3 기능별 Instruction 분류

1. Memory Integer Load/Store Summary

Mnemonic	Operation
LDA	Load Address
LDAH	Load Address High
LDBU	Load Zero-Extended Byte from Memory to Register
LDL	Load Sign-Extended Longword
LDL_L	Load Sign-Extended Longword Locked
LDQ	Load Quadword
LDQ_L	Load Quadword Locked
LDQ_U	Load Quadword Unaligned
LDWU	Load Zero-Extended Word from Memory to Register
STB	Store Byte
STL	Store Longword
STL_C	Store Longword Conditional
STQ	Store Quadword
STQ_C	Store Quadword Conditional
STQ_U	Store Quadword Unaligned
STW	Store Word

2. Control Instructions Summary

Mnemonic	Operation
BEQ	Branch if Register Equal to Zero
BGE	Branch if Register Greater Than or Equal to Zero
BGT	Branch if Register Greater Than Zero
BLBC	Branch if Register Low Bit Is Clear
BLBS	Branch if Register Low Bit Is Set
BLE	Branch if Register Less Than or Equal to Zero
BLT	Branch if Register Less Than Zero

3. Integer Arithmetic Instructions Summary

Mnemonic	Operation
ADD	Add Quadword/Longword
S4ADD	Scaled Add by 4
S8ADD	Scaled Add by 8
CMPEQ	Compare Signed Quadword Equal
CMPLT	Compare Signed Quadword Less Than
CMPLE	Compare Signed Quadword Less Than or Equal
CTLZ	Count leading zero
CTPOP	Count population
CTTZ	Count trailing zero
CMPULT	Compare Unsigned Quadword Less Than
CMPULE	Compare Unsigned Quadword Less Than or Equal
MUL	Multiply Quadword/Longword
UMULH	Multiply Quadword Unsigned High
SUB	Subtract Quadword/Longword
S4SUB	Scaled Subtract by 4
S8SUB	Scaled Subtract by 8

4. Logical and Shift Instructions Summary

Mnemonic	Operation
AND	Logical Product
BIC	Logical Product with Complement
BIS	Logical Sum (OR)
EQV	Logical Equivalence (XORNOT)
ORNOT	Logical Sum with Complement
XOR	Logical Difference
CMOVxx	Conditional Move Integer
SLL	Shift Left Logical
SRA	Shift Right Arithmetic
SRL	Shift Right Logical

5. Byte Manipulation Instructions

Mnemonic	Operation
LDBU/LDWU	Load byte/word unaligned
SEXTB/SEXTW	Sign-extend byte/word
STB/STW	Store byte/word

6. Memory Floating-Point Instructions

Mnemonic	Operation	Subset
LDF	Load F_floating	VAX
LDG	Load G_floating (Load D_floating)	VAX
LDS	Load S_floating (Load Longword Integer)	Both
LDT	Load T_floating (Load Quadword Integer)	Both
STF	Store F_floating	VAX
STG	Store G_floating (Store D_floating)	VAX
STS	Store S_floating (Store Longword Integer)	Both
STT	Store T_floating (Store Quadword Integer)	Both

7. Floating-Point Branch Instructions Summary

Mnemonic	Operation	Subset
FBEQ	Floating Branch Equal	Both
FBGE	Floating Branch Greater Than or Equal	Both
FBGT	Floating Branch Greater Than	Both
FBLE	Floating Branch Less Than or Equal	Both
FBLT	Floating Branch Less Than	Both
FBNE	Floating Branch Not Equal	Both

8. Floating-Point Operate Instructions Summary

Mnemonic	Operation	Subset
CPYS	Copy Sign	Both
CPYSE	Copy Sign and Exponent	Both
CPYSN	Copy Sign Negate	Both
CVTLQ	Convert Longword to Quadword	Both
CVTQL	Convert Quadword to Longword	Both
FCMOVxx	Floating Conditional Move	Both
MF_FPCR	Move from Floating-point Control Register	Both
MT_FPCR	Move to Floating-point Control Register	Both
ADDF	Add F_floating	VAX
ADDG	Add G_floating	VAX
ADDS	Add S_floating	IEEE
ADDT	Add T_floating	IEEE
CMPGxx	Compare G_floating	VAX
CMPTxx	Compare T_floating	IEEE
CVTDG	Convert D_floating to G_floating	VAX
CVTGD	Convert G_floating to D_floating	VAX
CVTGF	Convert G_floating to F_floating	VAX
CVTGQ	Convert G_floating to Quadword	VAX
CVTQF	Convert Quadword to F_floating	VAX
CVTQG	Convert Quadword to G_floating	VAX
CVTQS	Convert Quadword to S_floating	IEEE
CVTQT	Convert Quadword to T_floating	IEEE
CVTST	Convert S_floating to T_floating	IEEE
CVTTQ	Convert T_floating to Quadword	IEEE
CVTTS	Convert T_floating to S_floating	IEEE
DIVF	Divide F_floating	VAX
DIVG	Divide G_floating	VAX
DIVS	Divide S_floating	IEEE
DIVT	Divide T_floating	IEEE
FTOIS	Floating-point to integer register move, S_floating	IEEE
FTOIT	Floating-point to integer register move, T_floating	IEEE
ITOFF	Integer to floating-point register move, F_floating	VAX
ITOFS	Integer to floating-point register move, S_floating	IEEE
ITOFT	Integer to floating-point register move, T_floating	IEEE
MULF	Multiply F_floating	VAX
MULG	Multiply G_floating	VAX
MULS	Multiply S_floating	IEEE
MULT	Multiply T_floating	IEEE
SQRTF	Square root F_floating	VAX
SQRTG	Square root G_floating	VAX
SQRTS	Square root S_floating	IEEE
SQRTT	Square root T_floating	IEEE
SUBF	Subtract F_floating	VAX
SUBG	Subtract G_floating	VAX
SUBS	Subtract S_floating	IEEE
SUBT	Subtract T_floating	IEEE

9. Miscellaneous Instructions

Mnemonic	Operation
AMASK	Architecture Mask
CALL_PAL	Call Privileged Architecture Library Routine
ECB	Evict Cache Block
EXCB	Exception Barrier
FETCH	Prefetch Data
FETCH_M	Prefetch Data, Modify Intent
IMPLVER	Implementation Version
MB	Memory Barrier
RPCC	Read Processor Cycle Counter
TRAPB	Trap Barrier
WH64	Write Hint – 64 Bytes
WMB	Write Memory Barrier

10. VAX Compatibility Instructions

Mnemonic	Operation
RC	Read and Clear
RS	Read and Set

11. Multimedia (Graphics and Video) Support

Mnemonic	Operation
MINUB8	Vector Unsigned Byte Minimum
MINSB8	Vector Signed Byte Minimum
MINUW4	Vector Unsigned Word Minimum
MINSW4	Vector Signed Word Minimum
MAXUB8	Vector Unsigned Byte Maximum
MAXSB8	Vector Signed Byte Maximum
MAXUW4	Vector Unsigned Word Maximum
MAXSW4	Vector Signed Word Maximum
PERR	Pixel Error
PKLB	Pack Longwords to Bytes
PKWB	Pack Words to Bytes
UNPKBL	Unpack Bytes to Longwords
UNPKBW	Unpack Bytes to Words

제 2 절 GAS를 이용한 Alpha Assembly 언어 프로그래밍

앞의 Alpha Instruction을 이용해 Assembly 프로그램을 해보자.

2.1 컴파일 방법

1. Assembly 코드 작성

   가장 간단히 Assembly 언어를 만드는 방법은 다음과 같이 C언어에서 Assembly 언어를 추출해 내는 방법이다.
   다음은 hello.c 이다.

```
#include <stdio.h>
void main()
{
        printf("Hello World\n");
}
```

   다음은 C 소스에서 Assembler 소스를 얻는 방법이다.

```
% gcc -S hello.c
```

   다음은 이렇게 해서 얻는 Assembler 소스이다.

```
        .file   1 "hello.c"
        .version        "01.01"
        .set noat
gcc2_compiled.:
__gnu_compiled_c:
.section        .rodata
        .align 3
$C32:
        .ascii "Hello World\12\0"
.text
        .align 3
        .globl main
        .ent main
main:
        ldgp $29,0($27)
main..ng:
        lda $30,-16($30)
        .frame $15,16,$26,0
        stq $26,0($30)
        stq $15,8($30)
        .mask 0x4008000,-16
        bis $30,$30,$15
        .prologue 1
        lda $16,$C32
        jsr $26,printf
        ldgp $29,0($26)
$33:
        bis $15,$15,$30
        ldq $26,0($30)
        ldq $15,8($30)
```

```
        addq $30,16,$30
        ret $31,($26),1
        .end main
        .ident  "GCC: (GNU) 2.7.2.1"
```

2. GAS로 컴파일 하기

   그러면 hello.s라는 assembly 파일이 만들어 지는데, 이것을 GNU Assembler인 as로 Object 파일을 만든다.

   `% as -o hello.obj hello.s`

3. 실행 파일 만들기

   이제 Loader(ld)로 라이브러리를 합쳐 실행 파일을 만들어 주면 된다. 이때 gcc로 컴파일 해주면 자동으로 모든것을 알아서 해준다.

   `% gcc -o hello hello.o`

2.2 Alpha Assembly 언어 프로그래밍하기

앞의 C언어에서 만들어진 복잡한 hello.s를 간단하게 만들어 봄으로써 Alpha Assembly 언어를 익혀보자.

GAS로 프로그램을 할때 주의할 사항은 인텔과 같이 Instruction다음에 목적 Register와 소스 Register가 있지 않고 Instruction 다음읊努변됭egister와 목적 Register가 순서대로 되어 있는 AT&T 스타일로 되어 있다는 점이다.

다음은 hello.s 이다.

```
/* add.s
 * Alpha Assembly 프로그래밍 실습
 * 1998년 11월 24일 ssmLUG 정병수
 * Copyright (c) 1998 ssmLUG
 */
$msg:
.ascii "Hello World!\12\0"
.text
        .align 3
        .globl main
        .ent main
main:
lda $16, $msg
jsr $26, printf
ret $31, ($26), 1
        .end main
```

참고 서적

[1] **Alpha Architecture Handbook**, Version 4, COMPAQ

[2] **GNU Pro toolkit Manual**, http://www.cygnus.com/pubs/gnupro

[3] **Jan's Linux & Assember HomePage**, http://bewonner.dma.be/JanW/eng.html