Estructura de Computadores I 2014 - 2015

Memoria Práctica Final

Pablo Riutort Grande, 43185584W
pablo.riutort@gmail.com

Víctor Conrado Ladaria Lindem, 431869293H
victor.ladaria@gmail.com

24-05-15

Contents

1 Introducción

El objetivo de la práctica es emular a la máquina elemental denominada PEPA. En nuestro caso, la máquina cuenta con 4 eregistros de datos, un eregistro de estado y 12 instrucciones.
La máquina debe ser capaz de hacer el fetch, la decodificación y la ejecución de cada einstrucción emulando el comportamiento de la máquina elemental, por tanto, debe actualizar los flags en los casos que sea necesario y utilizar siempre el vector EPROG para capturar las einstrucciones y algunos operandos. En esta práctica hemos realizado algunas subrutinas para emular este comportamiento teniendo en cuenta las diferencias que hay entre la máquina elemental y el emulador EASy68k.

2 Desarrollo

Esta práctica se compone principalmente de una subrutina de decodificación (**DESPEP4R**) y einstrucciones que a su vez utilizan otra subrutinas auxiliares para emular el comportamiento de la PEPA.

2.1 Registros

Durante la ejecución de las einstrucciones, los registros más utilizados han sido **D4** y **D5**, estos suelen contener la dirección del eregistro al que se hace referencia para efectuar la einstrucción.
El registro D4 en especial ha sido utilizado para guardar el contenido del EIR y, como consecuencia, en la mayoría de subrutinas auxiliares como registro objetivo para realizar operaciones de decodificación (*BTST*). En la subrutina de flags D4 contiene el ESR.
El registro D5, al igual que D4, ha sido utilizado como un registro de propósito general en muchas subrutinas. Como generalmente este registro es utilizado para devolver el resultado de una einstrucción se ha utilizado en la subrutina de flags para actualizar el registro ESR en función de su constenido.
D2 es el registro utilizado para decodificacón de la einstrucción en la subrtuina **DESPEP4R**. A continuación adjuntamos una tabla donde se explica el uso frecuente de los demás registros.

Registro	**Uso**
D1	Uso del EPC
D2	Registro utilizado en la subrutina de librería DESPEP4R. Decodifica la einstrucción
D4 y D5	Regristros de propósito general en las subrutinas
A0	Registro utilizado para el EPROG
D3 y A2	Estos dos registros son utilizados para indexar JMPLIST

2.2 Subrutinas

2.2.1 DESPEP4R

Esta subrutina es llamada cada vez que se hace un fetch para decodificar la einstrucción.
DESPEP4R nos permite saber qué instrucción se ha codificado en el vector EPROG. Después del fetch de la einstrucción a ejecutar, llamamos a esta subrutina y mediante sucesivos *BTST* sobre el bit que nos interesa, vamos descartando las einstrucciones hasta que solo nos queda una. Luego, con este resultado, saltamos a una lista indexada que nos llevará a la einstrucción en cuestión. En este punto en el EIR tendremos el resto de información que nos hace falta para saber lo que realmente tiene que hacer nuestra einstrucción.

2.2.2 FLAGS

Existen 2 subrutinas que hacen referencia a los flags en esta práctica:

1. **FLAGS**: Es la subrtuina de flags completa. Comprueba los bits necesarios para activar los flags, esto se hace analizando el dato que se le ha pasado a la subrutina (D4) y haciendo *BTST n*

sobre el bit que nos interesa. Dependiendo del resultado de BTST sobre D5 haremos un *BSET n* siendo el n la posición del flag sobre el registro de estado y se lo pasaremos al eregistro ESR, emulando así un comportamiento de actualizaciones de flags.

2. FLAGNZ: El propósito de esta subrutina es actualizar únicamente los flags N y Z, ya que algunas einstrucciones lo especifican así. Básicamente lo que hace esta subrutina es llamar a FLAGS en un punto donde el flag C ya ha sido analizado.

2.2.3 TRUNCATE

La PEPA trabaja con registros de 12 bits, esta subrutina convierte los registros de 16 en 12, teniendo en cuenta que en el bit 12 se encuetra el valor del flag C. Esta subrutina tan solo actúa sobre los registros D4 y D5, pues son los más utilizados en las ejecuciones de las einstrucciones.

2.2.4 REG

Esta subrutina sirve para decodificar el registro al que hace referencia una einstrucción que tenga codificado el registro '*aa*'. Actúa sobre el registro D5, que contiene el EIR y deja en el registro de direcciones A4 el eregistro al que se hace referencia.

2.2.5 BREG

La subrutina BREG hace lo mismo que la subrutina REG, con la diferencia de que esta subrutina actúa sobre las '*bb*' de una einstrucción que tiene codificado un eregistro.
Utiliza el registro D4 y el registro de direcciones A3 para devolver el eregistro.

3 PRAFIN15

A continuación se expone una copia del código fuente de la práctica

```
*-----------------------------------------------------------
* Title      : PRAFIN15
* Written by : Pablo Riutort Grande y Victor Ladaria
* Date       : 27/5/15
* Description: Emulador de la PEPA4r
*-----------------------------------------------------------
    ORG    $1000

EPROG: DC.W $0700, $020F, $0606, $0C0B, $020E, $0605, $0C0B, $0418, $050D
       DC.W $0C0B, $0E07, $0601, $0010, $0F00, $0008, $0004, $0000

EIR:        DC.W    0   ;eregistre d'instruccio
EPC:        DC.W    0   ;ecomptador de programa
ER0:        DC.W    0   ;eregistre R0
ER1:        DC.W    0   ;eregistre R1
ER2:        DC.W    0   ;eregistre R2
ER3:        DC.W    0   ;eregistre R3
ESR:        DC.W    0   ;eregistre d'estat (00000000 00000ZNC)

START:                  ; first instruction of program

    CLR.W   EPC
FETCH:
    LEA.L EPROG,A0
```

```
    MOVE.W  EPC,D1
    ASL.W   #1,D1    ;multiplicamos el registro EPC por 2 (nos movemos words)
    ADD.W   D1,A0 ; con esto sabremos cual es la siguiente instruccion

    MOVE.W  (A0),EIR ; EIR ahora contiene la siguiente instruccion a ejecutar

    SUBQ.L  #2,A7 ;Reserva de espacio para variable local
    MOVE.W  EIR,-(A7) ;pasamos el registro EIR a la pila
    JSR DESPEP4R
    ADDQ.L  #2,A7 ;restauramos la pila
    MOVE.W  (A7)+,D3

    ;Nos preparamos para la ejecucion
    CLR.W   D4 ;registro auxiliar usado proximamente
    MULU #6,D3
    MOVEA.L D3,A2
    JMP JMPLIST(A2)

JMPLIST:
    JMP ESTO ;0
    JMP ELOA ;1
    JMP ECMP ;2
    JMP EADD ;3
    JMP ESUB ;4
    JMP ENAN ;5
    JMP EADQ ;6
    JMP ETRA ;7
    JMP ESET ;8
    JMP EJMZ ;9
    JMP EJMN ;10
    JMP EJMI ;11
    JMP EHLT ;12

UPEPC:
    ADDQ.W  #1,EPC
    BRA FETCH

;==== Subrutinas propias ====
;Subrutina de Flags
FLAGS:
    MOVE.L  D4,-(A7)
    MOVE.W  ESR,D4 ;Registro auxiliar para los flags

    BTST #12, D5
    BEQ NEG
    BSET #0,D4 ;actualizamos el flag c
NEG:
    BTST  #11, D5
    BEQ ZERO
    BSET #1,D4 ;actualizamos el flag n

ZERO:
    ;Comprobamos el flag z
```

```
    BTST #0, D5
    BNE CLEAR ;Si esta a 1 el flag z no debe ser actualizado
    BTST  #1, D5
    BNE CLEAR
    BTST  #1, D5
    BNE CLEAR
    BTST  #2, D5
    BNE CLEAR
    BTST  #3, D5
    BNE CLEAR
    BTST  #4, D5
    BNE CLEAR
    BTST  #5, D5
    BNE CLEAR
    BTST  #6, D5
    BNE CLEAR
    BTST  #7, D5
    BNE CLEAR
    BTST  #8, D5
    BNE CLEAR
    BTST  #9, D5
    BNE CLEAR
    BTST  #10, D5
    BNE CLEAR
    BSET #2,D4 ;Actualizamos el flag z

CLEAR:
    MOVE.W D4,ESR
    MOVE.L  (A7)+,D4

    RTS

FLAGNZ:
    MOVE.L D4,-(A7)
    BRA NEG

TRUNCATE:
;Preparamos los registros para el formato de la PEPA
    BCLR #15, D5
    BCLR #14, D5
    BCLR #13, D5
    ;BCLR #12, D5 este bit no hay que truncarlo, pues es el flag C

    BCLR #15, D4
    BCLR #14, D4
    BCLR #13, D4
;    BCLR #12, D4
    RTS

REG:
    ;subrutina para averiguar registros 'a'
    ;LO DEJA EN A4!
    MOVE.L D5,-(A7)
```

```
    MOVE.W EIR,D5

    BTST #3,D5
    BNE REGDOSTRES
    BTST #2,D5
    BNE REGUNO
    ;Sino, es el registro 0
    LEA ER0,A4
    BRA GER

REGUNO:
    LEA ER1,A4
    BRA GER

REGDOSTRES:
    BTST #2, D5
    BNE REGTRES
    ;Sino es 2
    LEA ER2,A4
    BRA GER

REGTRES:
    LEA ER3,A4

GER:
    MOVE.L (A7)+,D5
    RTS

BREG:
    ;subrutina para registros 'b'
    ;LO DEJA EN A3!
    MOVE.L D4,-(A7)
    MOVE.W EIR,D4

    BTST #1,D4
    BNE BREGDOSTRES
    BTST #0,D4
    BNE BREGUNO
    ;Sino es un 0
    LEA ER0,A3
    BRA BGER

BREGUNO:
    LEA ER1,A3
    BRA BGER

BREGDOSTRES:
    BTST #0,D4
    BNE BREGTRES
    ;Sino es 2
    LEA ER2,A3
    BRA BGER
```

```
BREGTRES:
    LEA ER3,A3

BGER:
    MOVE.L (A7)+,D4
    RTS

;==== Ejecucion de einstrucciones ====
EJMI:
    MOVE.W  EIR,D4
    AND #$00FF, D4
    MOVE.W  D4, EPC
    JMP FETCH

EJMZ:
    MOVE.W  ESR,D4
    BTST    #2,D4    ;Si Z=1 entonces, JMI
    BNE EJMI
    JMP UPEPC

EJMN:
    MOVE.W  ESR,D4
    BTST    #1,D4 ;Si N=1 entonces, JMI
    BNE EJMI
    JMP UPEPC

EJMC:
    MOVE.W  ESR,D4
    BTST    #0,D4
    BNE EJMI
    JMP UPEPC

ESTO:
    MOVE.W  EIR, D4
    AND #$00FF, D4
    MOVE.W  D4, A3
    MOVE.W  (ER1), (A3)
    JMP UPEPC

ELOA:
    MOVE.W  EIR,D4
    AND #$00FF, D4
    ASL.W   #1,D4
    MOVEA.L  D4, A3
    MOVE.W  EPROG(A3), (ER1)
    JMP UPEPC

ECMP:
    ;descubrir a que registros nos referimos
    JSR REG
    LEA ER0,A3
    MOVE.W  (A3),D4
    MOVE.W  (A4),D5 ;En A4 tenemos el registro deseado
```

```
    NEG.W   D5
    ;dejamos los registros como en la PEPA
    JSR TRUNCATE
    ;ahora podremos operar
    ADD.W   D4,D5
    ;actualizar los flags
    JSR FLAGS
    JMP UPEPC

EADD:
    LEA ER0,A3
    JSR REG
    MOVE.W  (A3),D4
    MOVE.W  (A4),D5
    JSR TRUNCATE
    ADD.W   D4,D5
    JSR FLAGS
    MOVE.W  D5,(ER0)
    JMP UPEPC

ESUB:
    ; A-B = A + ( B ) + 1 -> Esto lo que tiene que hacerse (PEPA)
    LEA ER0,A3
    JSR REG
    MOVE.W  (A3),D4
    MOVE.W  (A4),D5
    NEG.W   D5
    JSR TRUNCATE
    ADD.W   D4,D5
    JSR FLAGS
    MOVE.W  D5,(ER0)
    JMP UPEPC

ENAN:
    LEA ER0,A3
    JSR REG
    MOVE.W (A3),D4
    MOVE.W (A4),D5
    AND.W D4,D5
    JSR TRUNCATE
    NEG.W D5
    JSR TRUNCATE
    JSR FLAGNZ
    MOVE.W D5,(ER0)
    JMP UPEPC

EADQ:
    JSR BREG ;A3
    MOVE.W (A3), D4 ;bb
    MOVE.W EIR,D6

    BTST #3, D6
    BNE ADQNEG
```

```
        BTST #2, D6
        BNE ADQUNO
        ;es 0
        MOVE.W #0,D5
        BRA ADQEND

ADQUNO:
        ;es 1
         MOVE.W #1,D5
         BRA ADQEND

ADQNEG:
        BTST #2,D6
        BNE ADQNEGUNO
        ;es -2
        MOVE.W #2,D5
        NEG.W D5
        BRA ADQEND

ADQNEGUNO:
        ;es -1
        MOVE.W #1,D5
        NEG.W D5

ADQEND:
        ADD.W D4,D5
        JSR TRUNCATE
        JSR FLAGS
        MOVE.W D5,(A3)
        JMP UPEPC

ETRA:
        JSR REG ;Ra A4
        JSR BREG ;Rb A3
        ;Rb <- [Ra]
        MOVE.W (A4),(A3)
        MOVE.W (A3),D5
        JSR TRUNCATE
        JSR FLAGNZ
        JMP UPEPC

ESET:
        MOVE.W EIR,D4
        JSR BREG

        ;check c
        ;mscara para eliminar los 2 ultimos bits
        MOVE.W #$00FC,D5
        AND.W D4,D5
        ASR.W #2,D5 ;dividir por 2 nos dejara solo las c's
        MOVE.W D5,(A3)

        JMP UPEPC
```

```
EHLT:
    SIMHALT

;======Subrutina de decodificacion======

DESPEP4R:
    MOVE.L  D2,-(A7) ;guardamos el registro d2

    ;MOVE.W  EIR,D2
    MOVE.W  8(A7),D2
    BTST    #11,D2
    BNE PRIMERUNO

    BTST    #10,D2
    BNE NOSTORELOAD
    BTST    #9,D2
    BNE LOAD
    ;es un store
    MOVE.W #0,10(A7)
    BRA END

PRIMERUNO:
    BTST #9,D2
    BNE TERCERUNO
    BTST #8,D2
    BNE JMN
    ;es un jmz
    MOVE.W #9,10(A7)
    BRA END

TERCERUNO:
    BTST #8,D2
    BNE HALT
    ;es un jmi
    MOVE.W #11,10(A7)
    BRA END

NOSTORELOAD:
    BTST    #9,D2
    BNE TRASET
    BTST    #8,D2
    BNE ADQ
    BTST    #5,D2
    BNE SUBNAN
    BTST    #4,D2
    BNE ADD
    MOVE.W  #2,10(A7)
    BRA END

ADD:
    MOVE.W #3,10(A7)
    BRA END
```

```
SUBNAN:
    BTST    #4,D2
    BNE NAN
    MOVE.W  #4,10(A7)
    BRA END

NAN:
    MOVE.W  #5,10(A7)
    BRA END

ADQ:
    MOVE.W #6,10(A7)
    BRA END

TRASET:
    BTST #8,D2
    BNE SET
    ;es un tra
    MOVE.W  #7,10(A7)
    BRA END

SET:
    MOVE.W  #8,10(A7)
    BRA END

SUB:
    ;es un sub
    MOVE.W  #4,10(A7)
    BRA END

JMN:
    ;es un jmn
    MOVE.W  #10,10(A7)
    BRA END

LOAD:
    ;es un load
    MOVE.W  #1,10(A7)
    BRA END

HALT:
    ;es un halt
    MOVE.W  #12,10(A7)
    BRA END

END:
    MOVE.L  (A7)+,D2
    RTS

    END START
```