

UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA



Laboratorio N 1

Integrantes: Gabriel Gaete L.

Curso: Organización de Computadores

Sección 0-L-2

Profesor: Nestor González & Leonel Medina

22 de Abril de 2018

Tabla de contenidos

1. Introducción	1
2. Marco teórico	2
2.1. Conocimientos generales previos	2
2.2. Memoria Dinámica	2
2.3. Camino de datos o <i>datapath</i>	3
3. Desarrollo	4
3.1. Almacenamiento y estructuras para el manejo de datos	4
3.2. Identificación de instrucciones y validación de errores MIPS	5
3.3. Emulación de memoria	6
3.4. Escritura de archivos	6
4. Experimentos a realizar	7
4.1. Pruebas realizadas	7
4.1.1. Prueba 1	7
4.1.2. Prueba 2	7
4.2. Resultados obtenidos	8
4.2.1. Resultados prueba 1	8
4.2.2. Resultados prueba 2	8
4.3. Análisis de resultados	9
4.3.1. Prueba 1	9
4.3.2. Prueba 2	9
5. Conclusiones	10

1. Introducción

El presente informe pretende explicar cómo fue solucionado un problema que involucra el concepto de camino de datos, utilizando el lenguaje de programación estructurado C bajo el estándar de programación ANSI-C.

En la asignatura de “Organización de computadores”, se puede ver que una de las partes esenciales de los computadores corresponde al camino de datos, o “datapath”, el cual consiste en el camino que recorren las diferentes señales eléctricas a lo largo de un conjunto de unidades funcionales que permiten llevar a cabo la ejecución de las distintas instrucciones entregadas a un computador. Ante esto, surge la siguiente duda: ¿Qué unidades funcionales se activan al ejecutar un determinado conjunto de acciones?.

La principal motivación para llevar a cabo este laboratorio es comprender las unidades funcionales presentes dentro de un camino de datos, e identificar cuáles de estas se activan al ejecutar una determinada acción.

El objetivo principal de este informe es explicar como fue abordado el problema, para posteriormente llegar a una solución viable. Por otro lado, el objetivo principal del laboratorio es comprender e identificar cuáles son las instrucciones que dejan de funcionar al presentar errores de *stuck at 0* en determinadas líneas de control.

Las herramientas utilizadas en la implementación de la solución a este laboratorio son las siguientes:

1. Lenguaje de programación C
2. Uso de estructuras
3. Uso de memoria dinámica

2. Marco teórico

2.1. Conocimientos generales previos

El problema a resolver es el siguiente. A partir de dos archivos de entrada, uno con instrucciones en MIPS, el otro con listas de líneas de control, determinar, en dos archivos de salida, la traza del programa, junto con los valores que tomaría cada registro. Se debe tener en cuenta que las líneas Jump, Branch, Regwrite, Memread y Memwrite pueden presentar errores de *stuck at zero*.

A modo de manual, cabe mencionar que para una mayor facilidad a la hora de compilar y ejecutar los códigos fuentes del programa, se ha facilitado un *makefile*. Para su utilización, se tienen los siguientes comandos:

- *make all* permite la compilación el código fuente.
- *make run* permite ejecutar el programa.
- *make clean* permite eliminar el archivo ejecutable, luego de la utilización del programa.

Por último, cabe considerar que los archivos de entrada DEBEN estar en la misma carpeta que el programa ejecutable. Además, se recomienda que estos archivos sean de extensión .txt

2.2. Memoria Dinámica

Se refiere al uso de administración manual de memoria, es decir, dado que no se conoce inicialmente el tamaño que utilizará una variable, debe determinarse a medida que se requiera en el programa, lo que implica que no puede ser definida antes de compilar el programa. En el lenguaje de programación C se hace uso de esta técnica a través de las funciones *calloc*, *malloc*, *realloc* y *free*.

2.3. Camino de datos o *datapath*

El camino de datos corresponde a la ruta que sigue un dato de entrada dentro de un procesador, hasta que se obtiene un output. Está compuesto de unidades funcionales que pueden manejar información de forma conjunta. Comúnmente, dentro de un microprocesador, los caminos de datos están compuestos principalmente por *registros de instrucción*, una *unidad aritmético lógica*, *multiplexores*, una *unidad de control* y una *memoria*, tanto para instrucciones como para el almacenamiento de información.

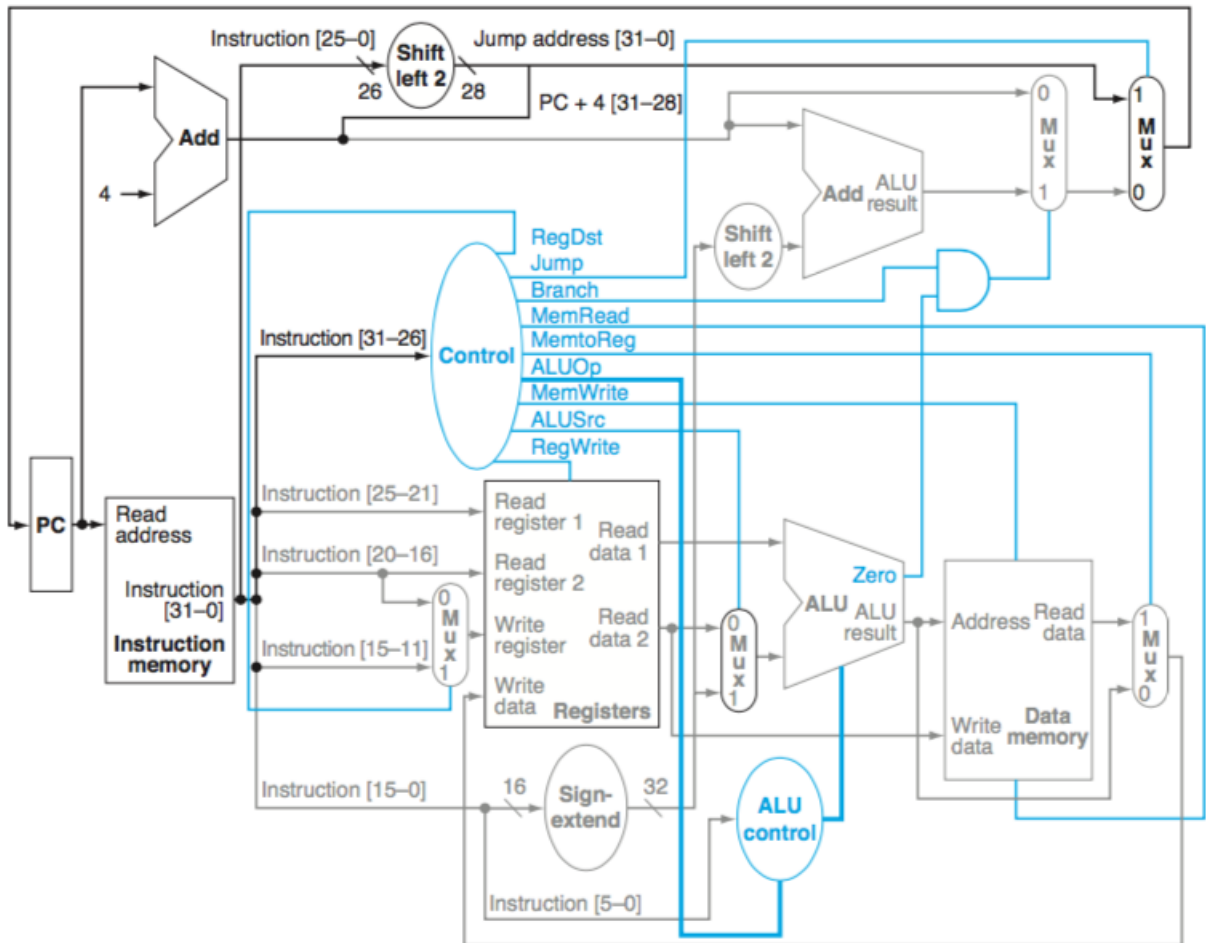


Figura 1: Ejecución de las instrucciones en las distintas etapas del datapath.

3. Desarrollo

En primer lugar, el problema presentado en el laboratorio fue dividido en varios sub-problemas, con el fin de ser resuelto por partes. Estos sub-problemas, fueron clasificados de la siguiente forma:

1. Almacenamiento y estructuras para el manejo de datos.
2. Identificación de instrucciones y validación de errores en MIPS.
3. Emulación de una memoria.
4. Escritura de archivos.

3.1. Almacenamiento y estructuras para el manejo de datos

El primero de los sub-problemas a resolver, se trató de encontrar una forma de almacenar los datos leídos de los archivos, puesto que el lenguaje de programación C no es un buen lenguaje para procesar texto. Los datos leídos, tanto en el archivo que se indican los errores de *stuck at 0*, como en el que se entregan las instrucciones MIPS, fueron almacenados dentro de listas enlazadas, en la que cada nodo de estas listas, corresponde a una línea de cada archivo.

```
typedef struct linesNode{
    char* line;
    struct linesNode* next;
}LinesNode;

typedef struct listOfLines{
    LinesNode* first;
    LinesNode* last;
    int length;
}ListOfLines;
```

Figura 2: Estructura que almacena las líneas de los archivos

Además de almacenar el contenido de los archivos, se hace necesario el uso de una estructura que permita almacenar las posibles restricciones que tenga el programa, es decir, las posibles limitaciones surgidas a partir de errores *Stuck at 0*. Se implementó una lista enlazada en la que se almacena el nombre de la línea de control, con su respectivo estado.

```

typedef struct restrictionsNode{
    char* instruction;
    char state;
    struct restrictionsNode* next;
}RestrictionsNode;

typedef struct restrictionsList{
    RestrictionsNode* first;
    RestrictionsNode* last;
    int length;
}RestrictionsList;

```

Figura 3: Estructura que almacena las líneas de control y sus estados

3.2. Identificación de instrucciones y validación de errores MIPS

Dado que la librería *string.h* entrega la posibilidad de comparar strings, al igual que la posibilidad de separar un string a través de un delimitador dado, es que la identificación de instrucciones se reduce a tomar la primera palabra de cada línea del archivo con instrucciones, y compararla con las instrucciones que el programa debe soportar. Una vez encontrada la función, sólo queda operar, ya sea sobre los registros, o sobre la memoria (siempre y cuando se pueda).

```

else if (strcmp(token, "beq") == 0){
    validateInstruction = 1;
    jump = 1;
    fprintf(f1, "%s\n", instruction);
    fprintf(f2, "\n%s", instruction);

    if (searchError(restrictions, "Branch") == 0){
        int firstRegister, secondRegister;
        token = strtok(NULL, " ");
        firstRegister = registers[searchRegister(token)];

        token = strtok(NULL, " ");
        secondRegister = registers[searchRegister(token)];

        if (firstRegister == secondRegister){
            jump = 2;
            token = strtok(NULL, " ");
            node = searchLabel(program, token);
        }
    }
}
}

```

Figura 4: Algoritmo que permite detectar instrucción 'beq' y comprobar si la línea de control respectiva no tiene error

3.3. Emulación de memoria

Dado que las funciones *lw* y *sw* trabajan con espacios de memoria, se hace necesario emular un espacio de memoria en la que se pueda guardar o cargar información. Para esto, la estrategia utilizada es la implementación de una matriz de números enteros, en la que se emula un espacio en la cual se pueden guardar hasta 100 números por dirección de registro.

```
memory = (int**)calloc(32, sizeof(int*));  
for(int i=0; i<32; i++) memory[i] = (int*)calloc(100, sizeof(int));
```

Figura 5: Matriz de números enteros que emula un espacio de memoria.

Cabe mencionar, que para el uso del programa, todos los registros son iniciados en 0, al igual que los espacios de memoria.

3.4. Escritura de archivos

Por último, el último sub-problema que se debió solucionar, fue la escritura de archivos, dado que con los sub-problemas anteriormente mencionados ya se puede generar una traza correcta de un programa básico en MIPS. Para esto, a medida que el programa interpreta las instrucciones entregadas por el archivo, y las opera en caso de ser necesario, se van imprimiendo tanto la traza como los valores de cada registro en los archivos de salida pertinentes.

```
if (validateInstruction != 0) for(int i=0; i<32; i++) fprintf(f2, " | %d", registers[i]);
```

Figura 6: Escritura del contenido de los registros dentro de un archivo

4. Experimentos a realizar

Para probar el programa realizado, se tienen los siguientes archivos de prueba:

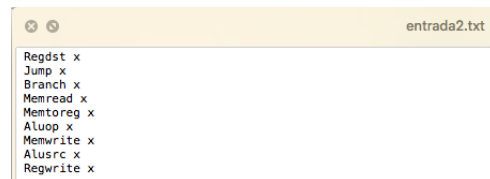
4.1. Pruebas realizadas

4.1.1. Prueba 1



```
addi $t0, $t0, 1
add $t1, $t0, $t0
beq $t1, $t0, EXIT
mul $t3, $t1, $t0
EXIT:
```

Figura 7: Archivo de entrada 1, contiene las instrucciones MIPS.



```
Regdst x
Jump x
Branch x
Memread x
Memtoreg x
Aluop x
Memwrite x
Alusrc x
Regwrite x
```

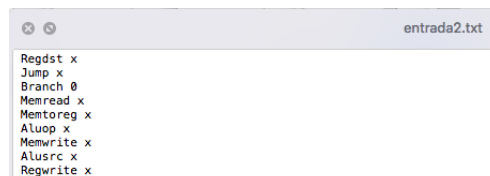
Figura 8: Archivo de entrada 2, contiene las líneas de control, en este caso, sin errores.

4.1.2. Prueba 2



```
addi $t0, $t0, 1
add $t1, $t0, $t0
sw $t0, 4($t1)
lw $t1, 4($t1)
beq $t1, $t0, EXIT
mul $t3, $t1, $t0
EXIT:
```

Figura 9: Archivo de entrada 1, contiene las instrucciones MIPS.



```
Regdst x
Jump x
Branch 0
Memread x
Memtoreg x
Aluop x
Memwrite x
Alusrc x
Regwrite x
```

Figura 10: Archivo de entrada 2, contiene las líneas de control, en este caso, stuck at 0 en branch.

4.2. Resultados obtenidos

4.2.1. Resultados prueba 1

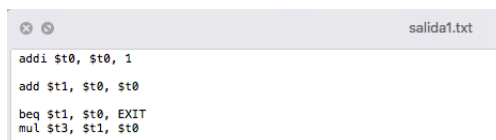


Figura 11: Archivo de salida1, contiene la traza del programa.

salida2.txt [Abrir con TextEdit](#)

```

INSTRUCTIONS    |s0|s1|s2|s3|s4|s5|s6|s7|s8|s9|s10|s11|s12|s13|s14|s15|s16|s17|s18|s19|s20|s21|s22|s23|s24|s25|s26|s27|s28|s29|s30|s31|s32|s33|s34|s35|s36|s37|s38|s39|s40|s41|s42|s43|s44|s45|s46|s47|s48|s49|s50|s51|s52|s53|s54|s55|s56|s57|s58|s59|s60|s61|s62|s63|s64|s65|s66|s67|s68|s69|s70|s71|s72|s73|s74|s75|s76|s77|s78|s79|s80|s81|s82|s83|s84|s85|s86|s87|s88|s89|s90|s91|s92|s93|s94|s95|s96|s97|s98|s99|s100|s101|s102|s103|s104|s105|s106|s107|s108|s109|s110|s111|s112|s113|s114|s115|s116|s117|s118|s119|s120|s121|s122|s123|s124|s125|s126|s127|s128|s129|s130|s131|s132|s133|s134|s135|s136|s137|s138|s139|s140|s141|s142|s143|s144|s145|s146|s147|s148|s149|s150|s151|s152|s153|s154|s155|s156|s157|s158|s159|s160|s161|s162|s163|s164|s165|s166|s167|s168|s169|s170|s171|s172|s173|s174|s175|s176|s177|s178|s179|s180|s181|s182|s183|s184|s185|s186|s187|s188|s189|s190|s191|s192|s193|s194|s195|s196|s197|s198|s199|s200|s201|s202|s203|s204|s205|s206|s207|s208|s209|s210|s211|s212|s213|s214|s215|s216|s217|s218|s219|s220|s221|s222|s223|s224|s225|s226|s227|s228|s229|s230|s231|s232|s233|s234|s235|s236|s237|s238|s239|s240|s241|s242|s243|s244|s245|s246|s247|s248|s249|s250|s251|s252|s253|s254|s255|s256|s257|s258|s259|s260|s261|s262|s263|s264|s265|s266|s267|s268|s269|s270|s271|s272|s273|s274|s275|s276|s277|s278|s279|s280|s281|s282|s283|s284|s285|s286|s287|s288|s289|s290|s291|s292|s293|s294|s295|s296|s297|s298|s299|s300|s301|s302|s303|s304|s305|s306|s307|s308|s309|s310|s311|s312|s313|s314|s315|s316|s317|s318|s319|s320|s321|s322|s323|s324|s325|s326|s327|s328|s329|s330|s331|s332|s333|s334|s335|s336|s337|s338|s339|s340|s341|s342|s343|s344|s345|s346|s347|s348|s349|s350|s351|s352|s353|s354|s355|s356|s357|s358|s359|s360|s361|s362|s363|s364|s365|s366|s367|s368|s369|s370|s371|s372|s373|s374|s375|s376|s377|s378|s379|s380|s381|s382|s383|s384|s385|s386|s387|s388|s389|s390|s391|s392|s393|s394|s395|s396|s397|s398|s399|s400|s401|s402|s403|s404|s405|s406|s407|s408|s409|s410|s411|s412|s413|s414|s415|s416|s417|s418|s419|s420|s421|s422|s423|s424|s425|s426|s427|s428|s429|s430|s431|s432|s433|s434|s435|s436|s437|s438|s439|s440|s441|s442|s443|s444|s445|s446|s447|s448|s449|s450|s451|s452|s453|s454|s455|s456|s457|s458|s459|s460|s461|s462|s463|s464|s465|s466|s467|s468|s469|s470|s471|s472|s473|s474|s475|s476|s477|s478|s479|s480|s481|s482|s483|s484|s485|s486|s487|s488|s489|s490|s491|s492|s493|s494|s495|s496|s497|s498|s499|s500|s501|s502|s503|s504|s505|s506|s507|s508|s509|s510|s511|s512|s513|s514|s515|s516|s517|s518|s519|s520|s521|s522|s523|s524|s525|s526|s527|s528|s529|s530|s531|s532|s533|s534|s535|s536|s537|s538|s539|s540|s541|s542|s543|s544|s545|s546|s547|s548|s549|s550|s551|s552|s553|s554|s555|s556|s557|s558|s559|s560|s561|s562|s563|s564|s565|s566|s567|s568|s569|s570|s571|s572|s573|s574|s575|s576|s577|s578|s579|s580|s581|s582|s583|s584|s585|s586|s587|s588|s589|s590|s591|s592|s593|s594|s595|s596|s597|s598|s599|s600|s601|s602|s603|s604|s605|s606|s607|s608|s609|s610|s611|s612|s613|s614|s615|s616|s617|s618|s619|s620|s621|s622|s623|s624|s625|s626|s627|s628|s629|s630|s631|s632|s633|s634|s635|s636|s637|s638|s639|s640|s641|s642|s643|s644|s645|s646|s647|s648|s649|s650|s651|s652|s653|s654|s655|s656|s657|s658|s659|s660|s661|s662|s663|s664|s665|s666|s667|s668|s669|s670|s671|s672|s673|s674|s675|s676|s677|s678|s679|s680|s681|s682|s683|s684|s685|s686|s687|s688|s689|s690|s691|s692|s693|s694|s695|s696|s697|s698|s699|s700|s701|s702|s703|s704|s705|s706|s707|s708|s709|s710|s711|s712|s713|s714|s715|s716|s717|s718|s719|s720|s721|s722|s723|s724|s725|s726|s727|s728|s729|s730|s731|s732|s733|s734|s735|s736|s737|s738|s739|s740|s741|s742|s743|s744|s745|s746|s747|s748|s749|s750|s751|s752|s753|s754|s755|s756|s757|s758|s759|s760|s761|s762|s763|s764|s765|s766|s767|s768|s769|s770|s771|s772|s773|s774|s775|s776|s777|s778|s779|s780|s781|s782|s783|s784|s785|s786|s787|s788|s789|s790|s791|s792|s793|s794|s795|s796|s797|s798|s799|s800|s801|s802|s803|s804|s805|s806|s807|s808|s809|s810|s811|s812|s813|s814|s815|s816|s817|s818|s819|s820|s821|s822|s823|s824|s825|s826|s827|s828|s829|s
```

Figura 12: Archivo de salida2, contiene los valores en registros.

4.2.2. Resultados prueba 2

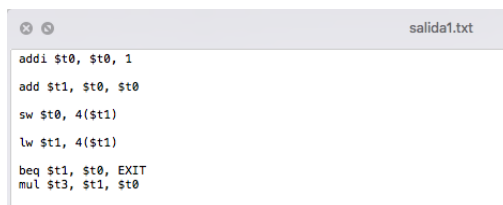


Figura 13: Archivo de salida1, contiene la traza del programa.

```

salida2.txt
Abrir con TextEdit

INSTRUCTIONS
addi $t0, $t0, 1
addi $t0, $t0, 2
sw $t0, 4($t1)
lw $t1, 4($t1)
beq $t1, $t0, EXIT
mul $t3, $t1, $t0

```

Figura 14: Archivo de salida2, contiene los valores en registros.

4.3. Análisis de resultados

4.3.1. Prueba 1

Para el caso de la primera prueba, se tiene que todas las líneas de control funcionan de manera normal, por tanto, no hay problemas a tener en cuenta a la hora de realizar una traza. Al analizar los archivos generados, tanto los valores de los registros como de la traza en general, los valores e ajustan a los resultados esperados.

4.3.2. Prueba 2

Para el segundo caso, se tiene que la línea de control *Branch* se encuentra con error de *Stuck at 0*, con lo cual, la instrucción *beq* no debiese ejecutarse correctamente. Al analizar los archivos generados, junto con los registros, se tiene que la instrucción anteriormente nombrada, debiese ejecutarse, puesto que los dos registros que compara son iguales, sin embargo, a pesar de esto, el salto a la etiqueta *exit* no se produce, tal como se esperaba al identificar que la línea de control *Branch* se encuentra en *Stuck at 0*.

5. Conclusiones

Concluyendo, en esta experiencia de laboratorio, se permite poner en práctica el concepto de *camino de datos*, junto a todo lo que esto conlleva, además, el programa permite probar el cómo un conjunto de instrucciones en MIPS fallan si determinadas líneas de control presentan errores.

Finalmente, puesto que se ha desarrollado el programa como se ha requerido, demostrando los conocimientos expuestos en la cátedra de la clase y se ha obtenido el resultado esperado, es posible concluir que se ha logrado cumplir con el objetivo principal de este laboratorio.